UNIVERSITY OF OSLO
Department of Informatics

Carry-Look-Ahead Adder in Multiple-Valued Recharge Logic

Cand. Scient. Thesis

Vidar Strønstad
Øverås

May 2005

Acknowledgements.

This thesis concludes my work for the Candidatus Scientiarum degree in Microelectronic Systems at the Department of Informatics, University of Oslo. My work was initiated in February 2003, and concluded May 2005.

I would like to thank my supervisor Yngvar Berg for accepting me as his student, for guidance and useful discussions. Furthermore I would like to pay tribute to Dag T. Wisland, Mats E. Høvin and Omid Mirmotahari for their assistance.

Next I would like to thank Espen Torstensen, Johannes Goplen Lomsdalen, Rene Jensen, Øivind Næss, Snorre Aunet, Olav Stanly Kyrvestad, Kjetil Meisal and Claus Limbodal for their contributions to this thesis, and also my fellow students at the laboratory, for scientific discussions and coffee breaks.

Last but not least, I would like to thank my family for their unconditional support and help, and Lena, for her patience, and understanding. Your support has been invaluable.

Abstract

In the last few decades, multiple-valued logics have been proposed as a possible alternative or enrichment to binary logic. Multiple-valued circuits replace the two states of binary logics with finite or infinite sets of values.

Many multiple-valued circuits, both current-mode and voltage-mode, have been published. To the best of our knowledge very few of these have had commercial success. However, multiple-valued recharge logics show great potential, reducing the number of transistors needed to perform a logic operation considerably when compared to binary logic. Present multiple-valued recharge adders use an inefficient carry-handling, and thus setting limitations of the number of bits that can be represented, as well as the max operation frequency. In this thesis a multiple-valued recharge adder making use of carry-look-ahead (CLA) is presented, addressing the carry-ripple problem. Furthermore, the presented multiple-valued CLA recharge adder is used in a proposed 16-bit CLA scheme. The CLA scheme is compared to a 16-bit multiple-valued recharge adder, regarding gate-delay, maximum operation frequency, and power consumption. Limitations of the proposed design is also presented.

Contents

Chapter 1

Introduction

1.1 Introduction

Binary logic has been the preferred logic for a long time. Like any other logic, binary logic has its advantages and disadvantages. The advantages are mainly accuracy and speed. The frequencies have increased as the transistor sizes have been downscaled. This, on the other hand, has increased the heat radiation caused by the transistors. Another problem is that today, internal routing occupies most of the chip area, as the number of transistors increase.

It is now understood and accepted that the limit for transistor size one day will be reached. When that time comes, binary logics will present few, if any, options on how to increase performance, as the number of transistors on a single chip will come to a stand still.

Multiple-valued logics (MVL) on the other hand offers the possibility to represent more than two logic values on a single line. Thus meaning that more operations can be carried out using fewer transistors. A result of this is that logic circuits that radiate far less heat can be designed. Many of the multiple-value floating gate circuits can not operate at the frequencies that binary logics operates on, but they compensate for this, by representing m logic values on a single line.

While MVL has a long history in literature, the big breakthrough considering industry has yet to come. MVL is most commonly used in memory designs, but this may change when transistor sizes reach a minimum.

Most of the proposed MV solutions are based on current-mode logics. The current-mode approach consumes significant power, due to static currents for each logic level. This has switched the attention to

voltage-mode multiple-valued logics. The voltage-mode approach, constructed using CMOS offers a design with less power consumption than the current-mode approach. Furthermore, voltage-mode MV full-adders have also been designed, and while using far less transistors than their binary counterparts, a need for better carry signal handling is addressed. The carry propagation of MV ripple adders restricts the operation frequency and the number of bits that can be represented. This again limits the circuits where adders are used. A MV ripple adder used in e.g. a decimator, limits both the operation frequency and the number of bits that can be represented. Therefore a solution to these problems has been addressed.

The scheme most commonly used for accelerating carry-propagation is called the carry-look-ahead (CLA) scheme [1]. The main idea of this scheme is to generate all incoming carries in parallel, an thus avoid waiting until the correct carry propagates from the adder where it has been generated. Figure 1.1 illustrates the basic idea of CLA. By implementing a MV adder using a CLA scheme, the delay of carry propagation will be reduced. This will lead to the opportunity to increase both the operation frequency and the number of bits that can be represented. This again will lead to an increase in performance for circuits using these adders.

All carry handling is carried out in the Carry-Look-Ahead Generator. $\mathbf{G}_i$ represents the internally-generated-carry signals. These signals do not depend on an external carry-in, only the original inputs. $\mathbf{P}_i$ on the other hand represents the sum where carry-out is **0** but where an external carry-in will trigger a carry-propagation. $\mathbf{G}_i$ and $\mathbf{P}_i$ are calculated in parallel, before an eventual carry-in is introduced.

CLA adders have been implemented using different approaches and technologies [2–6]. Also, a MV CLA adder designed in a 0.35 μm process, capable of operating at 400MHz has been reported [7]. The motivation for this thesis was to implement a voltage-mode multiple-valued semi-floating-gate adder using a carry-look-ahead scheme. The multiple-value floating gate CLA adder will be compared to the multiple-value floating gate full adder, in regard of gate-delay, max operation frequency, power-delay product and energy-delay product. Comments will also be made regarding the power consumption of the multiple-value floating gate Carry-Look-Ahead adder. For the solution designed for this thesis, summation is handled using multiple-valued signals. This results in an analog approach regarding the summation. One of the main reasons for this approach is that it has a more even use of power than binary solutions, which again results in a power effective solution. The carry handling on the other hand is carried out, using binary signals and binary gates, for fast shifting and propagation.

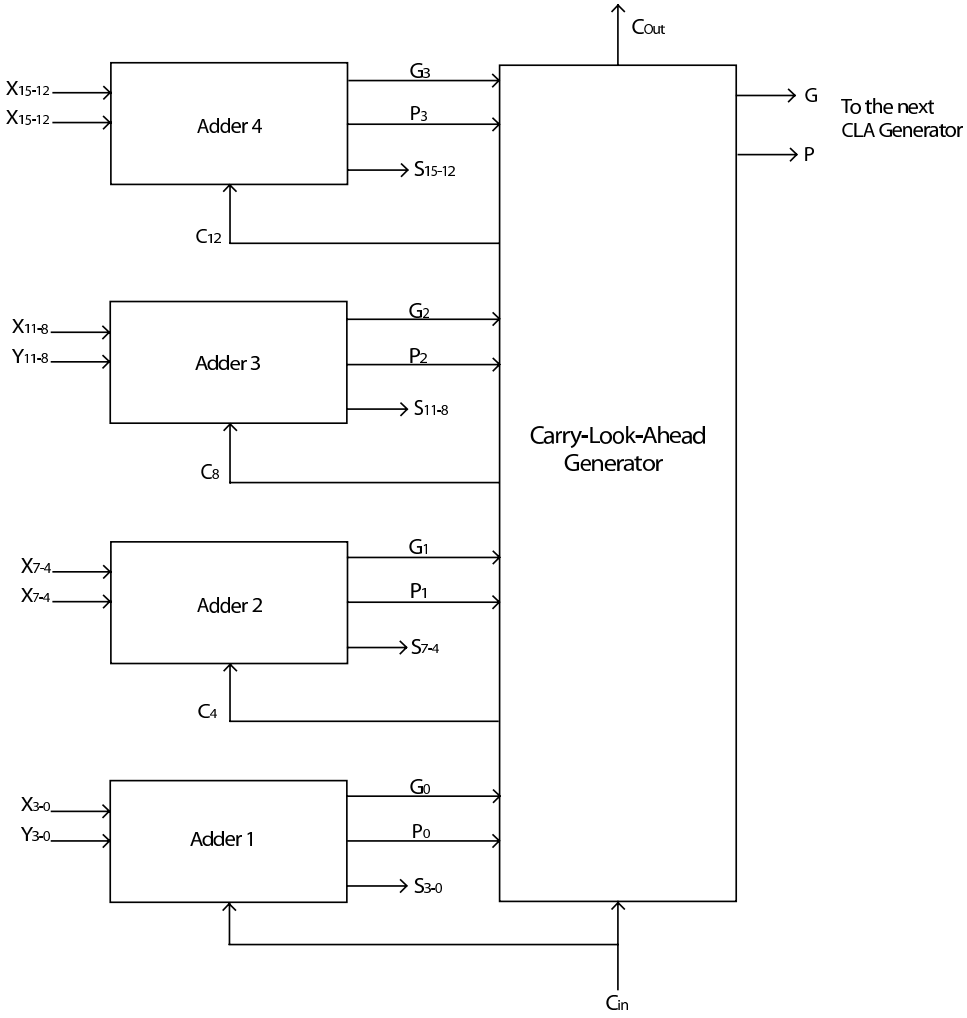

Figure 1.1: *A 16-bit two level carry-look-ahead adder. The notation* $\mathbf{X}_{3\text{-}0}$ *represents* $\mathbf{X}_3$, $\mathbf{X}_2$, $\mathbf{X}_1$, $\mathbf{X}_0$.

1.2 Overview of the thesis.

Chapter two present background information on multiple-valued recharge circuits. The presented circuits are published secondary literature, but the simulations are made from the layout of the prototype chip. The prototype chip is designed by the author of the thesis and fabricated using the AMS 0.35 μm CMOS process. Simulations were achieved using the *Spectre* simulator in *Cadence*. The author's contributions to multiple-valued recharge logics are the design of the prototype chip and the proposed carry-look-ahead scheme presented in chapter three. Considerations of the multiple-valued logic circuits are elaborated in chapter

four. Chapter five presents the conclusions and proposals for further work.

A listed outline of the thesis:

- **Chapter 1** gives a brief summary for the motivation of the thesis.
- **Chapter 2** presents the basic multiple-valued building blocks the prototype system is based upon. Also presented, is the MV Full-Adder.
- **Chapter 3** gives a thorough presentation of the MV Carry-Look-Ahead Adder. Furthermore a proposed carry-look-ahead scheme offers a great improvement over the MV full-adder considering gate-delay and frequency.
- **Chapter 4** includes considerations to the multiple-valued logics.
- **Chapter 5** presents a discussion and proposals for further work.

Chapter 2

Recharge Logic

In this chapter the basic circuits that are used throughout the thesis are introduced. Demonstrations on how they are constructed, and illustrations of their functionality is also provided. A full-adder based upon the basic circuits is also introduced. Further more, since multiple-valued logics need an interface that can communicate with existing binary systems, circuits that can translate binary to multiple-valued and vice versa are introduced.

2.1 The floating-gate inverter

The idea of using capacitive coupled inputs on a FG-inverter was introduced in 1992 by Shibata [8–10]. This FG-inverter has later been known as the neuron-MOSFET, since it resembles the behaviour of the "neurons" in a living body [11].

The floating-gate (FG) inverter [12] is composed using standard MOS-transistors and capacitors, as presented in Figure 2.1a. The input signal of an FG inverter is capacitive coupled, as opposed to a binary inverter. For this reason the FG inverter can have M input signals. Using inputs that are capacitive coupled, means that the output is calculated by the voltage on the floating-gate. A small change on the floating-gate, will result in a large voltage change on the output of the inverter. With M input signals, the voltage on the floating-gate represents several input signals, as will be described later.

In order to obtain a consistent and desired function, the FG circuits need to be initialized or programmed. On a local scale, the ultra-violet (UV) FG inverter has been the object of much research [13–17]. The UV FG inverter is programmed using ultra violet light. The downside with the UV

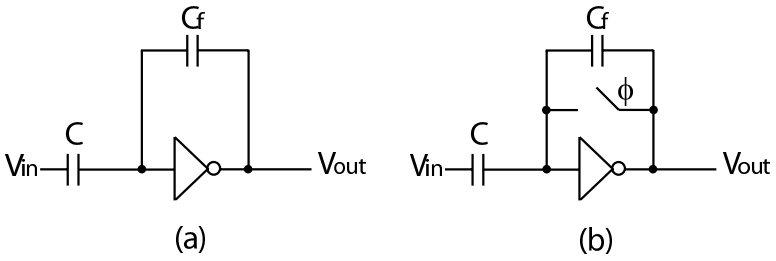

Figure 2.1: *The Illustration shows both the SFG binary inverter (a) and the SFG MV inverter (b).*

scheme, is that it is unpredictable. Although UV programming is defined as a "once and for all" approach, some circuits suffer from leakage and thus needs to be reinitialized often. Other circuits on the other hand, hold the initialized charge stable for a longer period of time.

In 1992, a clock-controlled FG-inverter was proposed [18]. The introduction of a clock-controlled floating-gate, means that the inverter is recharged every clock-period to a known voltage level, often $V_{\mathrm{dd}}/2$. Since the inverters are recharged each clock period, the leakage on the floating-gate is minimized. The clocked recharge results in a semi-floating-gate (SFG). By introducing a capacitive coupling between the semi-floating-gate and the output of the SFG inverter, the opportunity to represent more than two logical levels appear. This capacitor is called "feedback capacitor" (C_f). The MV SFG inverter is described in Figure 2.1b.

Since the SFG inverter with C_f can represent more than two logical levels, noise margin becomes a much more critical issue than for digital inverters. Noise margin becomes important when the number of bits represented on the single line increases. MV SFG logics have also been criticized for the limitations regarding high frequencies. Since several bits are represented on a single line, the noise margin implicates the rate of which bits are transferred. On the other hand, since a single line can represent several bits, more bits can be calculated within one clock period.

Another beneficial aspect of MV SFG logic, is that power consumption is static. With a static power consumption, the logics do not need to be designed to tolerate large spikes, caused by clock switching. Although the power consumption is static, the average power consumption is slightly higher than for binary gates. Further discussion on this topic is presented in chapter four.

The MV SFG inverter can also be used as a latch, and this will be explained in the next section.

2.2 The multiple-valued semi-floating-gate latch

As mentioned the multiple-value semi floating gate inverter can also be used as a small memory element, a latch [19]. If the SFG inverter, phased ϕ (in a single phased clocking scheme), is followed by a SFG inverter, phased $\overline{\phi}$, the second inverter will operate as a latch. Figure 2.2 illustrates the principle of the SFG latch.

The binary SFG latch does not invert the signal, but delays it 1/2 clock phase, and refreshes it as well. The signal propagation is illustrated in Figure 2.3. When the first inverter is recharging (setting the value to $V_{dd}/2$) the second inverter is evaluating. This implies that the first evaluation of the second inverter (the latch) cannot be determined, since it is evaluating a previous voltage level at output $\mathbf{out_1}$. When the first inverter is recharging (the clock is ϕ), the output $\mathbf{out_1}$ is $V_{dd}/2$. The recharge period is followed by the evaluation period (the clock is $\overline{\phi}$), where the input (**In**) changes to either V_{dd} or V_{ss}. The output $\mathbf{out_1}$ switches to either V_{ss} or V_{dd} respectively, inverting the input signal (**In**). During this period, the second inverter is recharging. Notice that the output $\mathbf{out_2}$ is $V_{dd}/2$. When the first inverter switches over to recharging, the output $\mathbf{out_1}$ switches to $V_{dd}/2$, while the previous output of $\mathbf{out_1}$ (either V_{ss} or V_{dd}) is evaluated by the second inverter. The ouptut $\mathbf{out_2}$ is now output $\mathbf{out_1}$ latched. It is worth noticing that the voltage changes (ΔV) are $V_{dd}/2$ for this latch. Furthermore, the difference between recharge and evaluate is defined by ΔV.

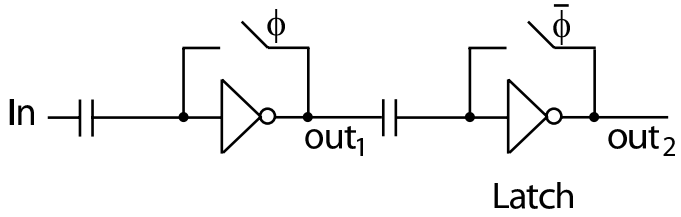

Figure 2.2: *The binary recharge latch, the latch is the second inverter.*

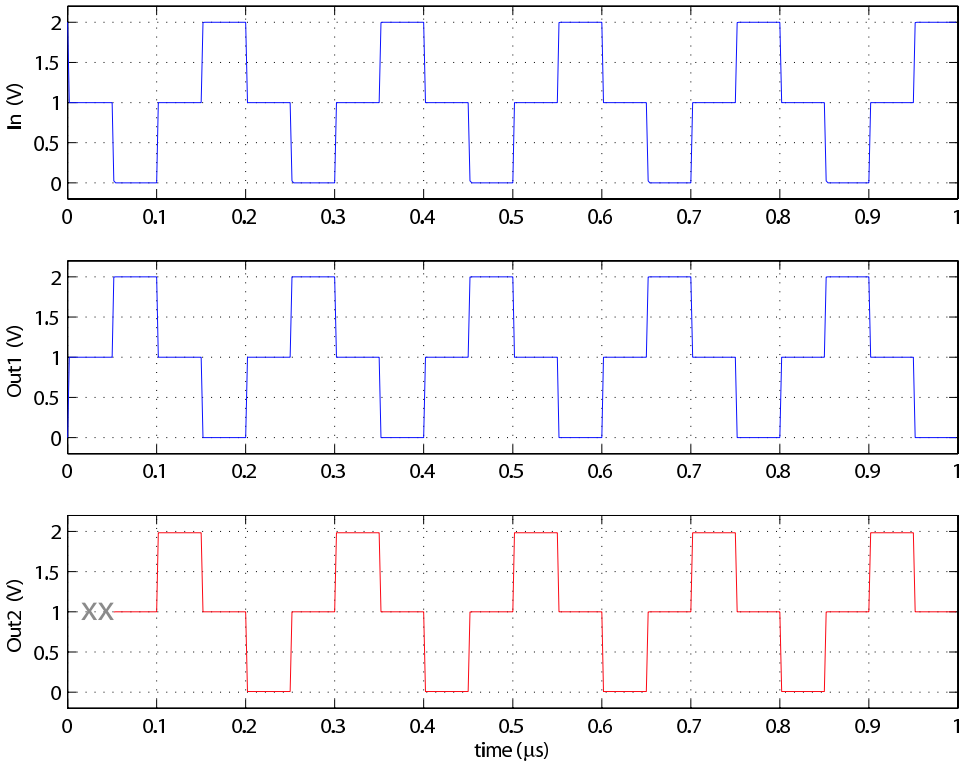

Figure 2.3: *Signal propagation through the latch in Figure 2.2* $\mathbf{out_2}$ *is* $\mathbf{out_1}$ *delayed 1/2 clock period. The frequency is 10MHz.*

The multiple-valued recharge latch can be constructed in the same manner, only adding feedback capacitors to the inverters. This latch is presented in Figure 2.4. The signal propagation closely resembles the binary SFG latch. The difference is that ΔV is not necessarily $V_{dd}/2$, but can be different voltage changes according to the logic value presented.

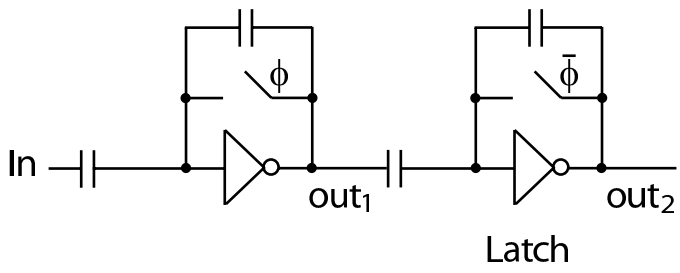

Figure 2.4: *The binary recharge latch. The actual latch is the second inverter.*

The gain of the multiple-valued recharge inverter can be given by the equation

$$G = \frac{C_i}{C_f + C_{gd} + \frac{C_T}{A}} \tag{2.1}$$

Where $\mathbf{C_i}$ is the capacitor coupled to the input, $\mathbf{C_f}$ the feedback capacitor, $\mathbf{C_{gd}}$ the gate-drain capacitance, $\mathbf{C_T}$ the total capacitance and **A** is given by

$$A = \frac{gm}{go} \tag{2.2}$$

In an inverter based amplifier, A is negative, thus leading to G being negative. Furthermore, given that

$$A \to \infty \tag{2.3}$$

And also given

$$C_{dg} \to 0 \tag{2.4}$$

This will leave the following simplified equation of G

$$G = \frac{C_i}{C_f} \tag{2.5}$$

Furthermore, to be able to move the negative value out of **G**, the absolute value of the original equation of **G** is used. The original equation is used for a more correct representation of G

$$G = \left| \frac{C_i}{C_f + C_{gd} + \frac{C_T}{A}} \right| \tag{2.6}$$

This equation can be used to illustrate Figure 2.4. Now, given that the previous circuit to the first inverter is clocked ϕ, $\mathbf{Out_1}$ can be expressed

$$Out_1(n) = -G_1 \cdot In(n) \tag{2.7}$$

This leads to

$$Out_2(n+1) = G_2 \cdot Out_1(n) \tag{2.8}$$

These expressions lead to

$$Out_2(n) = -G_1 \cdot G_2 \cdot In(n-1) \tag{2.9}$$

In this thesis, a one phased clocking scheme is used, and the circuits are therefore sensitive to clock skew. The gain of the circuits is also sensitive to the value of **A**. The smaller **A** is, the smaller the gain. A small value of **A**, can imply that the value of $\mathbf{g_o}$ is large, due to channel length modulation, also called Early effect. This can be caused by short transistors.

The presented MV latch has also been proposed used to implement static memory cells [20]. Furthermore, a multiple-valued static-static memory has been proposed for synaptic storage [21] [22].

2.3 The capacitor

The most important components of the presented circuits are the capacitors, therefore it is important to take a look at some of the possible implementation solutions. Capacitances can be extracted between almost any layer and doped area, in different degrees. The overlapping area can be calculated with the formula $C = \epsilon \cdot A/d$. In this formula ϵ is the dielectric constant for the oxide between the layers, A is the overlap area of the layers, and d is the distance between the layers.

There are different ways to implement capacitances, although only a few will be mentioned here. For this thesis the first solution presented has been used. The reason for this, is that this solution presents the least complicated modules. The other solutions are mentioned to present an overview over other approaches.

1) **Interpoly capacitor:** *A capacitor is implemented using two layers of polysilicon on top of each other. Many MOS technologies that are used to implement analog circuits have two layers of polysilicon [23].*

2) **Finger capacitor:** *Depends on the design rules for each individual process. It is typically used in the absence of a double-poly process, and is constructed by poly and metal layers. The* 0.12μ *STM process can be mentioned as an example of processes using this technique. In this process, capacitors are designed using metal* 1, 2, 3, 4, 5 *and* 6*, together with active and poly. The metal layers are fingered to increase the capacitance.*

3) **Coupled capacitance:** *A coupling capacitance is usually seen as a parasitic capacitance. Coupling capacitance is either the result of two layers interacting, fringing capacitance, or a result of two different layers crossing, with the thick oxide separating the layers.*

To minimize the effect of nonlinearity and parasitic capacitances, stable, well matched capacitors are needed. As the radix (logic levels) increases, this becomes even more important. When using multiple-valued logics, it is important to implement large enough capacitors to prevent domination of non-linearity and parasitic capacitances.

2.4 Binary-to-recharge-binary-converter

To be able to translate a binary signal to a multiple-valued signal, a recharge level needs to be included to the signal. For this purpose the binary-to-recharge-binary-converter (also called Auto-Zero or AZ) can be used [24]. When the reset is equal to 0, meaning that the circuit is not recharging, it operates precisely as a regular clocked inverter. On the other hand, while it is recharging, the output is driven to the recharge state defined by $\mathbf{V_{Out}} = V_{dd}/2$. Figure 2.5 shows the schematic view of the AZ.

The AZ is used to include a recharge level to digital signals before these are introduced to the recharge circuits or MV recharge circuits. It is worth noticing that the binary signal is inverted through the AZ, and inverted again when converted to a multiple-valued signal. The reason for this, is that the systems used, except the latch, are inverting systems. A binary signal introduced to the AZ and a following recharge inverter will therefore be presented logically correct, only with a recharge level included.

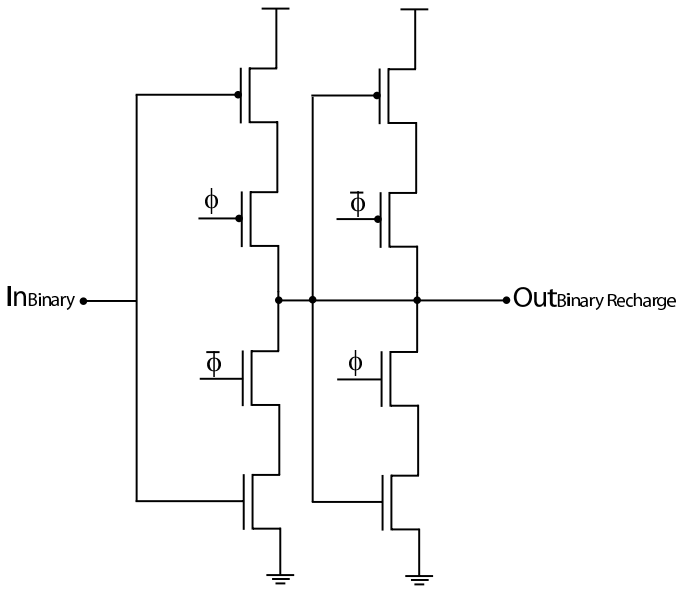

Figure 2.5: *Auto-Zero circuit used to include a recharge period to binary signals.*

2.5 Binary-to-multiple-valued-converter and Multiple-valued-to-binary-converter

Since multiple-valued logic require an interface that can communicate with existing binary systems, circuits that can translate binary to multiple-valued and vice versa are needed. The binary-to-multiple-valued-converter (BMVC) combined with the AZ, represents a fully functional translator of binary to multiple-valued signals.

The BMVC [24] is a multiple-valued inverter with m input signals, as shown in Figure 2.6. The BMVC is a MV SFG inverter with several inputs, connected to the floating-gate using representative sized capacitors. The capacitors are used to weigh the input signals according to significance. The truth table in Figure 2.7 and the layout simulation of the BMVC in Figure 2.8 verifies the design.

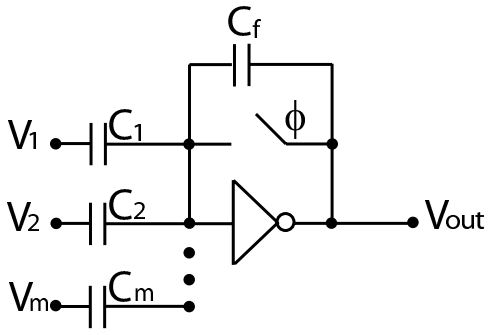

Figure 2.6: *The design shows the Semi-Floating-Gate binary-to-multipled-value-converter, with m input signals.*

V1	V2	Vout	$\overline{\text{Vout}}$
0	0	0	3
1	0	1	2
0	1	2	1
1	1	3	0

Figure 2.7: *The truth table of the 3-bit BMVC.*

The circuit used to down-convert multiple-valued signals to recharge binary, is called the multiple-valued-to-binary-converter (MVBC). This circuit is a bit more complex than the BMVC previously described. A suggestion for design of the MVBC is presented in [20]. The MVBC used in this thesis is shown in Figure 2.9. This MVBC has few fan-ins, which is

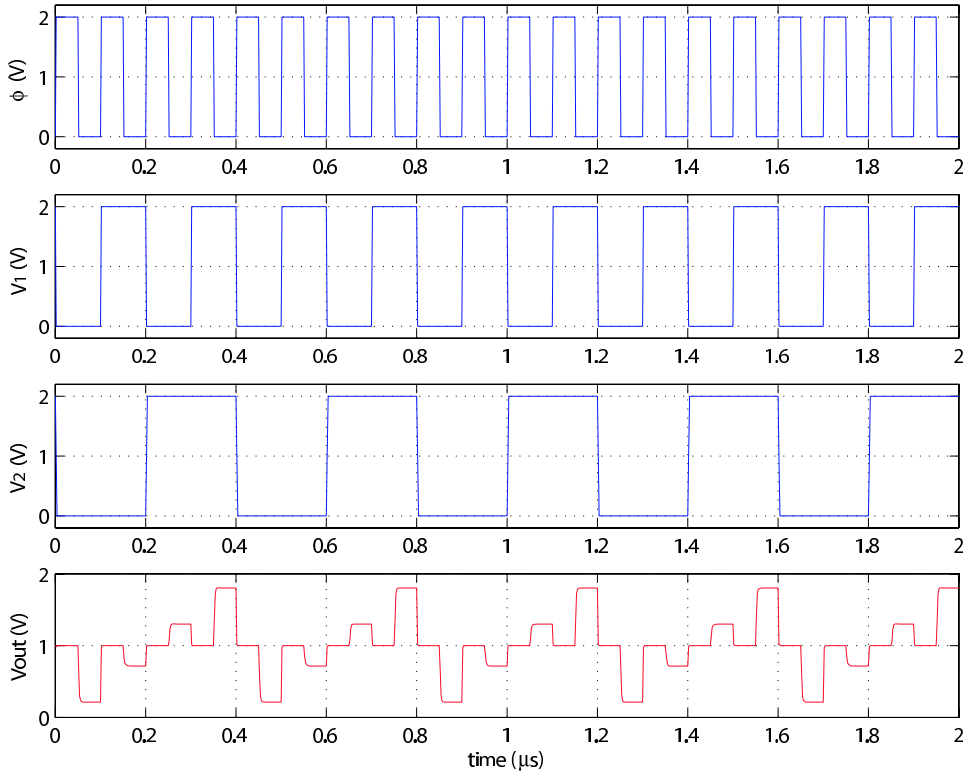

Figure 2.8: *Layout simulation of the BMVC, which includes the Auto-Zero. The AZ includes a recharge level of $V_{dd}/2$ to the binary inputs. The recharge-binary signals are used as inputs on the BMVC. The frequency is 10MHz.*

desirable considering the difficulties of matching the capacitors.

The signal propagation of the MVBC is as follows: The most significant bit (**MSB**) is determined by whether or not the multiple-valued signal is higher than $V_{dd}/2$ (inverter 1). The output signal of inverter 1 is the inverted (**MSB**) (node **X**). In order to obtain the logically correct value, the signal in node **X** needs to be inverted (inverter 2). When adding together the input signal and node ($\overline{\textbf{MSB}}$) (inverter 3), the result is the remaining bits of the input signal node **Y**. Inverter 4 is used to invert node **Y** to obtain the logical correct **LSB**. The layout simulation of the MVBC in Figure 2.10 and the corresponding truth table in Figure 2.11.

The basic MV circuits presented earlier, combined with the peripheral circuits described in this section offer the opportunity to design MV SFG circuits that can communicate with a binary world. The described circuits can also be used as building blocks for more complicated multiple-valued structures. In the next section a multiple-value semi-floating-gate full-adder is presented.

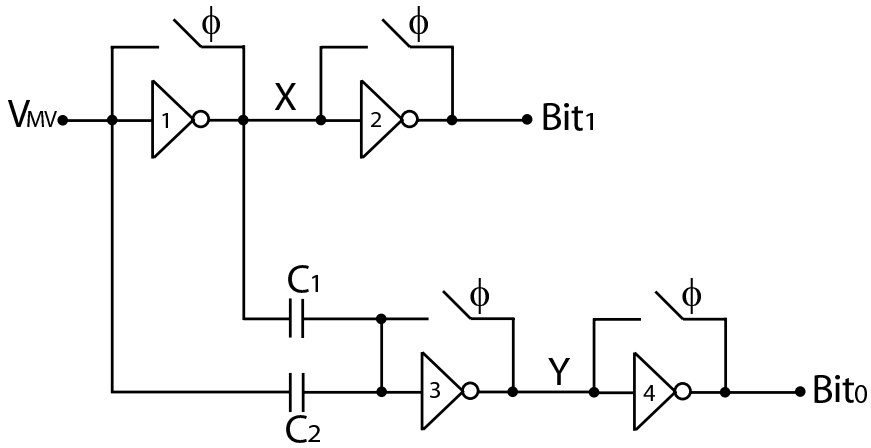

Figure 2.9: *The Multi-value-to-binary converter.The capacitance values are $C_1 = 4/3C$ and $C_2 = 7/3C$, where C is the unit capacitor.*

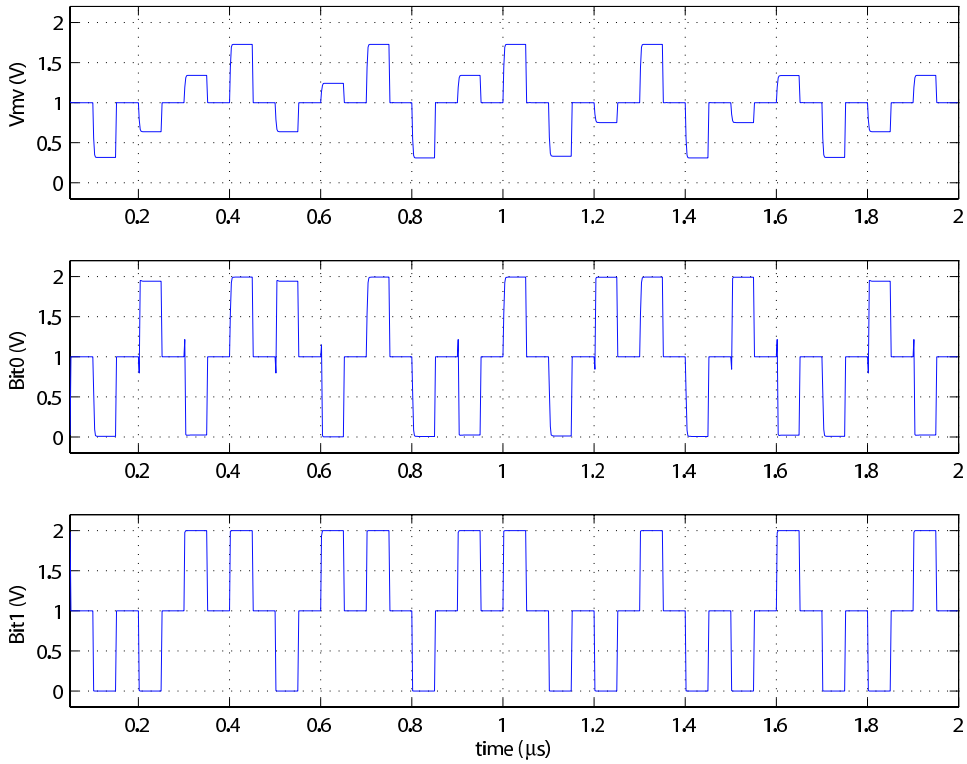

Figure 2.10: *Simulation of layout, showing how the MVBC operates. The R4 input results in two a two-bit output, the output bits are represented with separate wires. In this simulation the carry in is logic* **0**. *The frequency is 10MHz.*

2.6 Adder

Like most full-adders, the MV full-adder has three input signals, e.g. $\mathbf{X}_i$, $\mathbf{Y}_i$ and $\mathbf{C}_i$, and two output signals, e.g. $\mathbf{C}_o$ and **Sum**. The multiple valued signals to be summed are represented by $\mathbf{X}_i$ and $\mathbf{Y}_i$ while $\mathbf{C}_i$ represents the carry-in. The output signal **Sum** represents, as the name suggests, the sum, while $\mathbf{C}_o$ represents the carry-out of the adder. The radix of $\mathbf{C}_o$

V_{MV}	Bit1	Bit0
0	0	0
1	0	1
2	1	0
3	1	1

Figure 2.11: *The truth table of the 2-bit MVBC.*

is equal to the radix of $\mathbf{C}_i$, namely radix-2. Furthermore the radix of the **Sum** should not exceed the radix of the input signals, $\mathbf{X}_i$ or $\mathbf{Y}_i$.

The adder presented here is shown in Figure 2.12 [25]. The first inverter performs the addition of the input signals. Since $\mathbf{X}_i$ and $\mathbf{Y}_i$ have a larger radix than $\mathbf{C}_i$, these signals are weighed with larger capacitances than $\mathbf{C}_i$. Node (**Z**) represents the inverted **Sum** in a radix of 2-R, where **R** is the radix of $\mathbf{X}_i$ and $\mathbf{Y}_i$. Inverter 2 determines the carry-out signal ($\mathbf{C}_o$), by inverting node (**Z**). The values of node (**Z**) below $V_{dd}/2$ are inverted to binary-recharge signals above $V_{dd}/2$, and thus representing the carry-out ($\mathbf{C}_o$). Inverter 3 determines the output **Sum**, of a radix equal to $\mathbf{X}_i$ and $\mathbf{Y}_i$. Since node (**Z**) represents the inverted **Sum**, and inverter 3 inverts this, while down-converting the radix, the output **Sum** is logically correct. The MV adder is demonstrated in Figure 2.13.

When compared to a standard single-bit binary full-adder [26] shown in Figure 2.14, the simplicity of the MV SFG full-adder becomes apparent. This binary full-adder needs 28 transistors to represent one bit, thus implying that it needs 56 transistors to be able to represent 2-bit. The MV SFG full-adder on the other hand needs 12 transistors (3 inverters and 6 clock transistors) to represent the same number of bits.

The multiple-valued recharge adder presented here is one of many different approaches in design of MV adders. This particular adder sums the input signals and the carry-in simultaneously using one MV inverter, but this is not always desirable. The disadvantage with the MV full-adder described here, becomes obvious when several adders are combined to represent more bits. A 16-bit MV full-adder will include eight 2-bit MV full-adders, through which the carry signal must ripple. This is briefly illustrated in Figure 2.15.

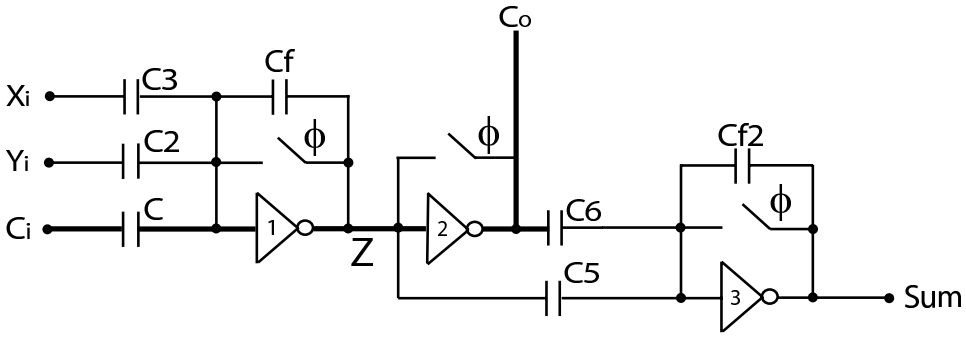

Figure 2.12: *The MV SFG full-adder. The capacitance values are* $C = C_{f2} = C_{min}$, $C_2 = C_3 = (R-1)C$, $C_f = (2R-1)C$, $C_6 = \frac{R}{(R-1)}C$ *and* $C_5 = \frac{(2R-1)}{(R-1)}C$, *where* C *is the unit capacitor.*

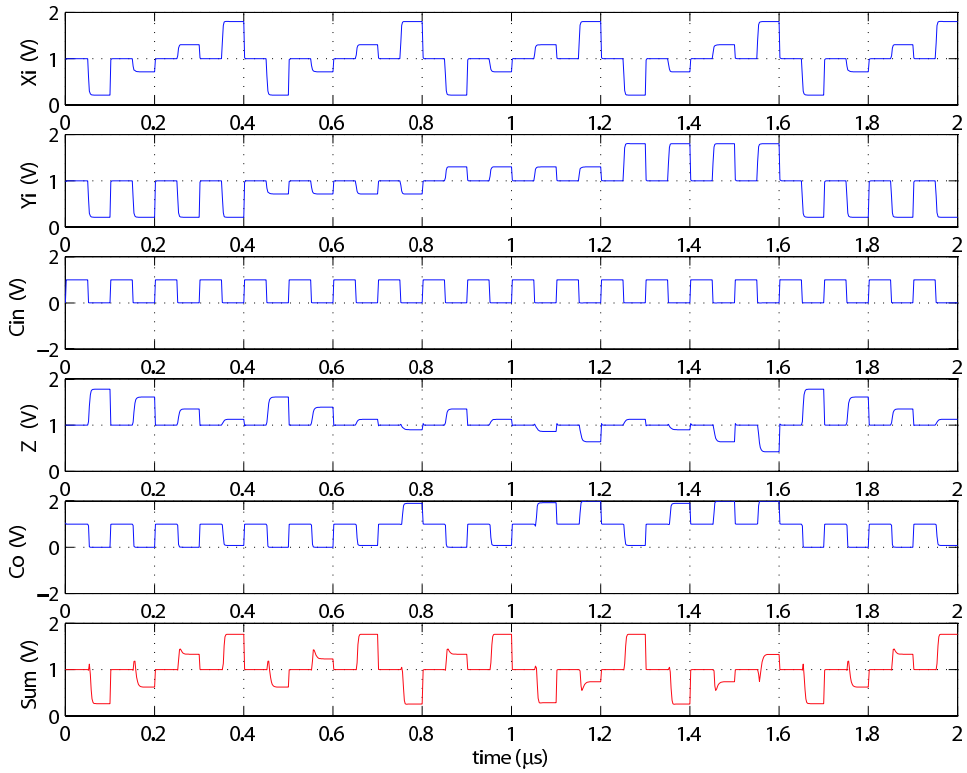

Figure 2.13: *The simulation demonstrates the MV SFG full-adder. The two R4 inputs are generated by two BMVC's. The simulation is performed on the layout designed, all parasitic capacitances are included. The frequency is 10MHz.*

For each two-bit MV adder, the carry signal ripples through two inverters. This implies that a carry signal must ripple through sixteen inverters for a cascaded 16-bit MV full-adder. The result is a significant gate-delay affecting the ripple of the carry signal. In Figure 2.16 the ripple delay of 16-bit MV full-adder is illustrated. The adders are stacked as shown in Figure 2.15. Also a a table providing the transistor sizes used, is presented in Figure C.2.

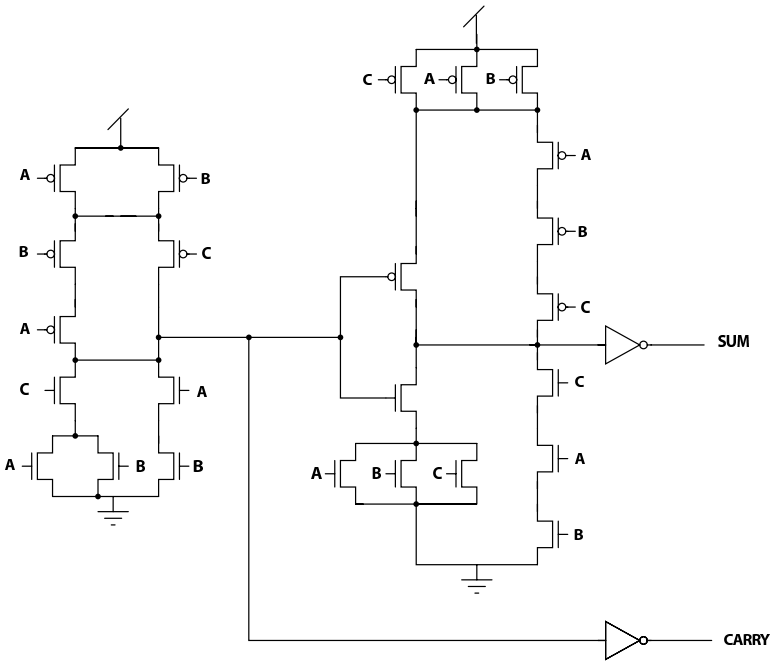

Figure 2.14: *Single bit Binary Full-Adder.*

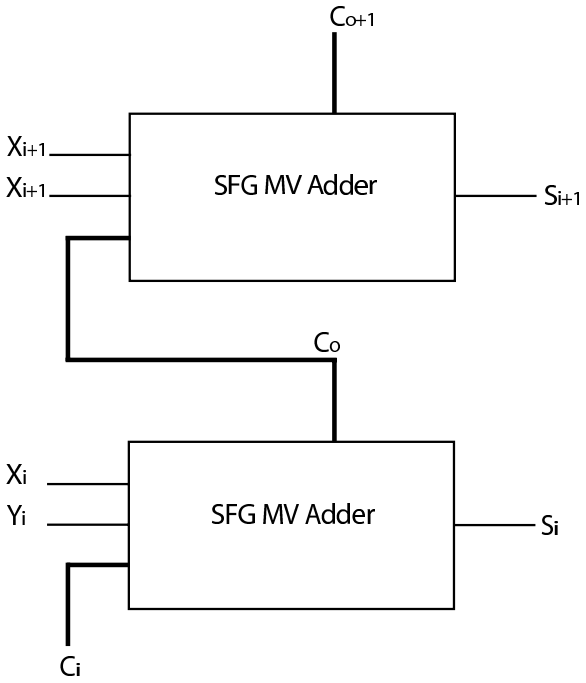

Figure 2.15: *The Figure illustrates the ripple of the carry signal when using eight 2-bit SFG MV Adders cascaded, though only two adders are depicted here.*

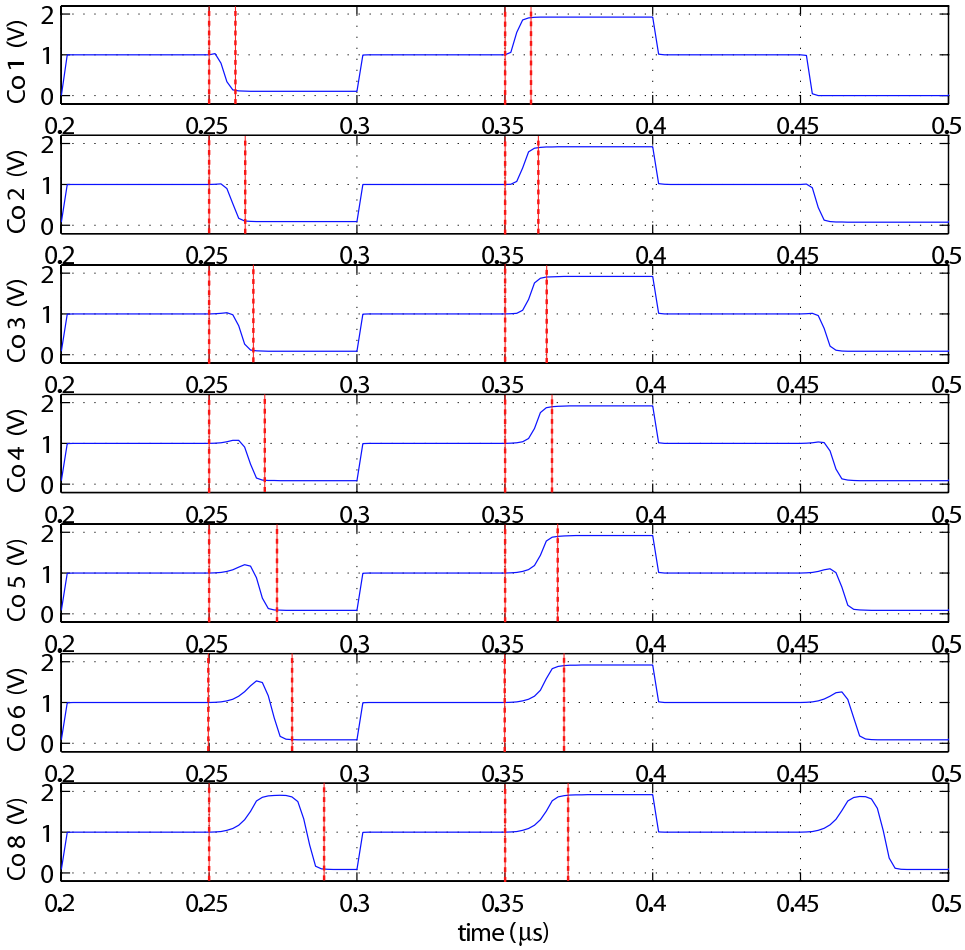

Figure 2.16: *Simulation illustrating the ripple delay of the carry signal through eight cascaded 2-bit MV SFG full-adders. The nMOS and pMOS transistors are 0.6 µm and 3.05 µm wide respectively, while both have minimum length equal to 0.35 µm.*

In the figure, $\mathbf{C}_o$ 1 represents the carry-out signal of the first 2-bit adder-element, $\mathbf{C}_o$ 2 the second, and so on. The limitations of the design are apparent in the carry-out ($\mathbf{C}_o$ 8)of the 16-bit adder. The worst-case delay of this signal is when the carry-out is **0**. This gate-delay is approximately 40ns, and limits the operation clock frequency for the implemented version of the adder to maximum 10MHz. The gate-delay of a positive carry-out is approximately 22ns, also quite a significant delay. If the gate-delay caused by one inverter is represented by $\Delta_{\mathbf{G}}$, where **G** represents the gate of an inverter, the total gate-delay for the carry signal of the 16-bit MV SFG full-adder will reach $16\Delta_{\mathbf{G}}$. Reducing this significant gate-delay as much as possible, would result in faster and more flexible

adders.

Figure 2.16 also shows that the actual voltage change Δ_V for the recharge periods happen in parallel, without any gate-delay. This implies that the recharge transistors do not impose any delay or limitations to the MV circuits at this frequency. Therefore it is the carry rippling that imposes the largest hindrance for the system.

2.7 Summary

In this chapter the basic multiple-value circuits, and the usage of these has been presented. The circuits described in this chapter can be used in larger circuits, or adapted to give other beneficial solutions. Furthermore the limitations of the presented MV adder has been pointed out. A possible solution to this problem would be to introduce a carry-look-ahead scheme, to decrease the delay on the carry signal. A MV carry-look-ahead full-adder is presented in the next chapter, along with a 16-bit carry-look-ahead scheme with a minimum delay regarding the carry signal.

Chapter 3

The multiple-valued Carry Look Ahead Adder

In this chapter a carry-look-ahead scheme for multiple-valued logics is proposed. Also, the prototype multiple-valued carry-look-ahead adder used in this scheme is presented and verified by measurements. The prototype adder makes use of the basic components described in the previous chapter, but offers a much more effective carry-handling. Furthermore, the proposed carry-look-ahead scheme is explained using the prototype chip and additional circuits.

The prototype adder designed for this thesis is a modified version of the recharge adder described in chapter two. For faster carry-handling, the prototype adder actually consists of two recharge adders and additional gates. One of the adders calculates the correct **Sum**, while the other adder, along with additional gates, is used for carry calculation. The fact that two MV adders and additional gates are used, more than doubles the size of the adder compared to the recharge adder. On the other hand, the prototype adder offers a much more efficient carry-handling than the recharge adder, in terms of speed. By decreasing the gate-delay of the carry signal, the adder can operate at faster frequencies. The decrease of gate-delay also opens the opportunity to design adders that represent more than sixteen bits, while still using 2-bit adders as building blocks.

3.1 The Prototype Carry-Look-Ahead Full-Adder

For the prototype adder designed, the peripheral components described in chapter two were used for the interface communicating with a binary world. This was done for two reasons. By surrounding the prototype adder with the peripheral components, it would be possible to test if the

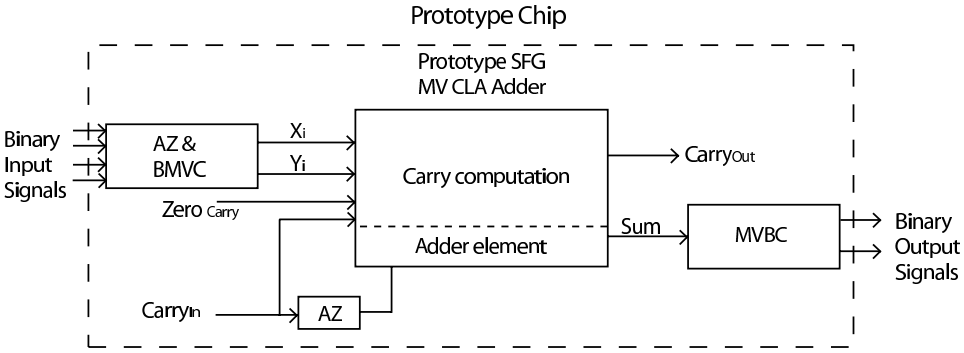

Figure 3.1: *The design shows the schematic view of the SFG MV Carry-Look-Ahead Adder, with the peripheral interface to communicate with binary components.*

system would work with binary components. The second reason was signal instrument control scripts. By using these peripheral components, only binary signals are needed. These are less complicated to script in Matlab than multiple-valued signals. Figure 3.1 shows an overview of the prototype adder with the peripheral components used.

Due to the more effective carry-handling, the MV carry-look-ahead (CLA) full-adder is more complex in its design than the MV full-adder. To be able to move the carry-handling out of the adder and in to the Carry-Look-Ahead Generator, whilst being able to compute the correct output **Sum**, it has been mentioned that the adder actually consists of two adder elements. The first to be described, is the element used to calculate the internally generated carry signal and the carry-in sensitive **Sum**, $\mathbf{G}_i$ and $\mathbf{P}_i$ respectively.

The adder element used to calculate $\mathbf{G}_i$ and $\mathbf{P}_i$, is a modified version of the MV full-adder presented in chapter two. It has been mentioned that $\mathbf{G}_i$ and $\mathbf{P}_i$ should preferably be binary signals due to effectiveness.

As the intention is to calculate $\mathbf{G}_i$ and $\mathbf{P}_i$ the input-signals have to be summed without the inclusion of a carry-in. This is done in the same manner as with the MV full-adder, only that the carry-in signal added here is logic “0”, hence the name $\mathbf{Zero}_{\text{Carry}}$. The functionality of the $\mathbf{Zero}_{\text{Carry}}$ is to obtain correct radix on the summation of the inputs. The described design is illustrated in Figure 3.2.

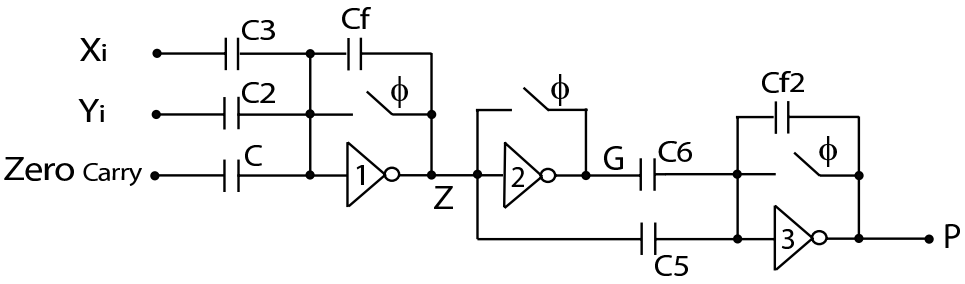

Figure 3.2: *The design shows the first section of the schematic view of the MV Carry-Look-Ahead Adder.*

The summed inputs are represented in node **Z**. As with the MV full-adder the **Sum** in node **Z** is inverted. Due to the inclusion of the $\mathbf{Zero}_{\text{Carry}}$, the **Sum** in node **Z** ranges from **0** to **6**. This implies that when the carry signal is calculated in "inverter 2", the output of the inverter represents the generated-carry, **G**. In other words, **G**=1 when ($\mathbf{X_i} + \mathbf{Y_i} > \mathbf{R}$ of $\mathbf{X_i}$ or $\mathbf{Y_i}$) and **0** otherwise. This is verified by layout simulation in Figure 3.3.

The next step is to further calculate **P**. The desired representation of **P**, is when the **Sum** = the radix of $\mathbf{X_i}$ and $\mathbf{Y_i}$, and **G** is **0**. This combination will generate a carry-out if there is a carry-in. Although at this stage, **P** represents all combinations of the sum. The design is verified in the truth table in Figure 3.4. The values marked blue represent the carry-sensitive value of **P**.

The ideal representation of **P**, is a binary signal, where the **Sum** (**3**) and the generated carry **G** (**0**) is represented by the logic value **1**, and all other instances produce a logic **0**. To achieve this, a binary recharge latch has been implemented, illustrated in Figure 3.5.

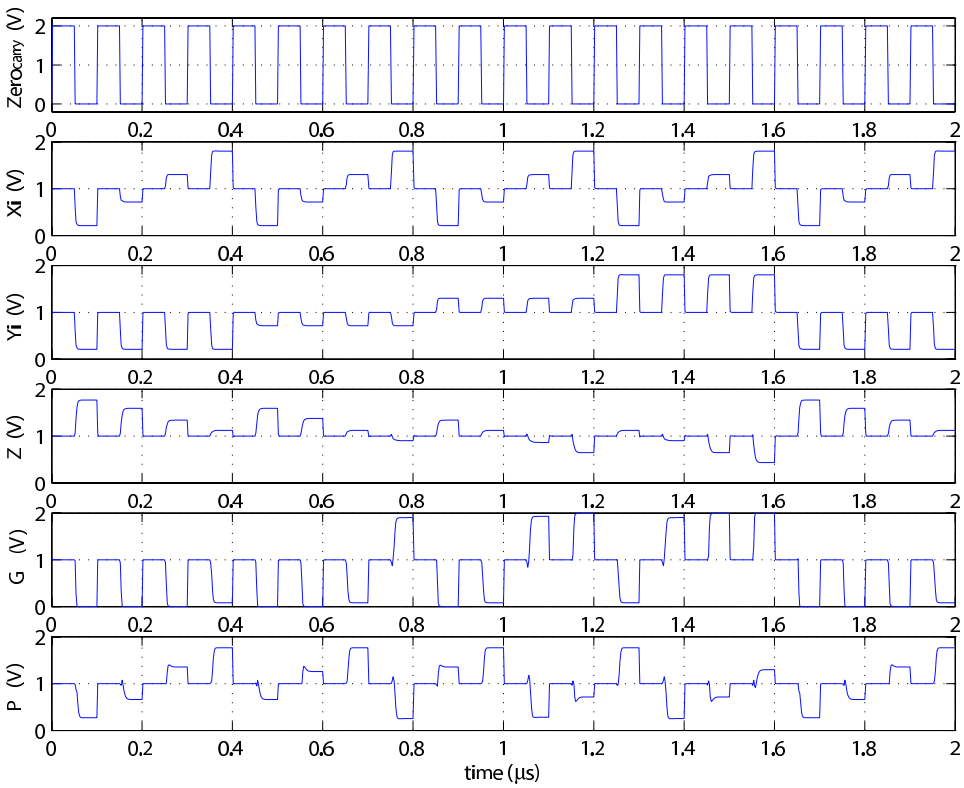

Figure 3.3: *Layout simulation of the carry element of the adder. The* $\mathbf{Zero}_{\text{Carry}}$ *is equal to the clock signal used. The* $\mathbf{Zero}_{\text{Carry}}$, X_i *and* Y_i *are summed, resulting in the R8 signal* **Z**. *Node* **Z** *is used to determine generated carry,* **G**, *which in turn is summed with the latter to give the R4 representation of* **P**. *The clock frequency is 10 MHz.*

$\overline{Z}$	G	P
0	0	0
1	0	1
2	0	2
3	0	3
4	1	0
5	1	1
6	1	2

Figure 3.4: *The truth table for* **G** *and* **P**.

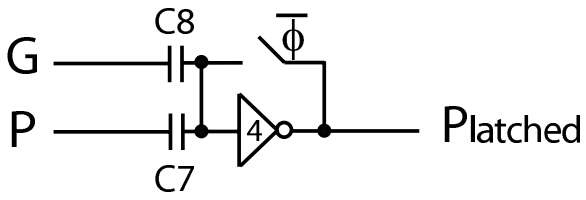

Figure 3.5: *The latch used to down convert the carry-sensitive* **P**. **G** *and the radix-4 representation of* **P** *are used as inputs. The signals are weighed* **C** *(***C8***) and* **2C**(**2**/**1.6**) *(***C7***) respectively. The reason for weighing* **C7** *with* (**2**/**1.6**) *is to adjust the weighing according to the voltage swing of the signals. The previous circuits are clocked* ϕ*, thus the latch needs to be* $\overline{\phi}$*.*

The weighing of the signals ensure the functionality desired for the circuit, except that some of the other combinations of input values generate a logic **1** as output as well. These other input values of **G** and **P** of radix-4 are (**1** and **1**), or (**1** and **2**) respectively. This is illustrated truth table in Figure 3.6 and by the measurements in Figure 3.7.

Measurements show that the output values of the latch are not always binary-recharge. This may be caused by unmatched capacitors. Parasitic capacitances may also contribute to this effect, though these are dependent on the size of the capacitors. Although this might appear as a problem, the gate following (a NAND gate) only needs the values to be above or below $V_{dd}/2$ to obtain correct functionality. Another thing that needs to be commented, is the measured radix-4 presentation of **P**. The inaccuracy of the signal is partially a result of inaccuracy in the previous stages. Reasons for these inaccuracies are elaborated in chapter four.

G	P	Platched
0	0	0
0	1	0
0	2	0
0	3	1
1	0	0
1	1	1
1	2	1

Figure 3.6: *Truth table for the latch used to down convert* **P**.

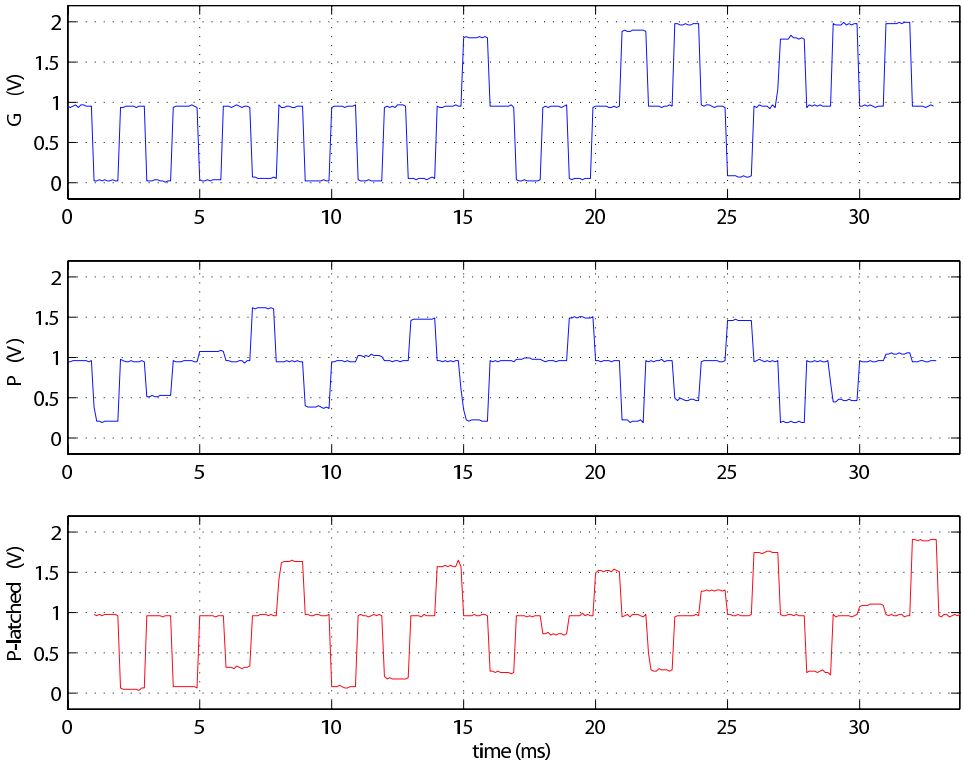

Figure 3.7: *Measurement of the latch used to obtain* $\mathbf{P_{latched}}$. *The generated carry* **G**, *is latched with the* **P**, *resulting in* $\mathbf{P_{latched}}$. *Although the output,* $\mathbf{P_{latched}}$, *is supposed to be recharge-binary, not all input combinations achieve this. This may be caused by unmatched capacitors. The frequency is 500Hz.*

Due to the weighing of the input signals to the latch, other values of **P** and **G** than **3** and **0** respectively can trigger a carry-out. A carry-out triggered by these combinations will cause bit error, and thus they need to be filtered out. For this purpose a NAND gate has been used. The NAND gate and its inputs are illustrated in Figure 3.8.

As the output of a NAND gate is logic **0** only when all inputs are **1**, and **0** for all other inputs, an inverted and latched generated-carry ($\overline{\mathbf{G}}$) is used as input along with $\mathbf{P_{latched}}$. Inverter **5** in Figure 3.8 inverts the generated-carry **G**, while inverter **6** is used to latch the inverted-generated-carry ($\overline{\mathbf{G}}$). By inverting **G**, the undesired combinations resulting in **P** being carry-in sensitive can be eliminated. This is illustrated in the truth table in Figure 3.9. Furthermore measurement verifications of this design are found in Figure 3.10.

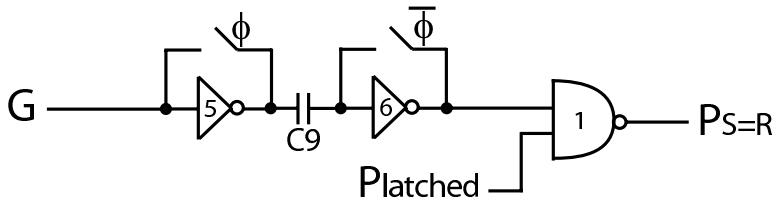

Figure 3.8: *The NAND gate used to isolate the carry-sensitive sum of* **P**. $\overline{\mathbf{G}}$ *and* $\mathbf{P_{latched}}$ *are used as inputs.*

Since the NAND gate is a binary-gate, the inaccuracy of $\mathbf{P_{latched}}$, is of no concern. As the measurements illustrate, as long as the inputs are correct (above or below $V_{dd}/2$) the output values of the NAND-gate are logically correct. It is worth noticing that the output of the NAND gate is pulled down to 0.5V when the input is $V_{dd}/2$ (recharge level). This is due to the fact that both the nMOS and pMOS transistors are turned on at this voltage, resulting in a voltage level influenced by both the nMOS and pMOS transistors.

As previously described, $\mathbf{P_{latched}}$ represents the carry-in sensitive signal **P**, thus this signal is not in phase with the generated carry **G**. To be able to use both signals in the Carry-Look-Ahead Generator, both must be in phase. To achieve this, a binary recharge latch has been used on **G**.

With the output of the NAND gate, and the latched generated carry **G**, the carry-handling element of the adder used in the proposed carry-look-ahead scheme is completed. Although for the prototype MV CLA adder, the remaining gates used to calculate the correct carry-out were also

C_{in}	$\overline{Z}$	$\overline{G_{latched}}$	$P_{latched}$	$P_{S=R}$
1	0	1	0	1
1	1	1	0	1
1	2	1	0	1
1	3	1	1	0
1	4	0	0	1
1	5	0	1	1
1	6	0	1	1

Figure 3.9: *Truth table for the NAND gate used to determine* **P**. *The carry-in sensitive value of* **P** *is marked in blue.*

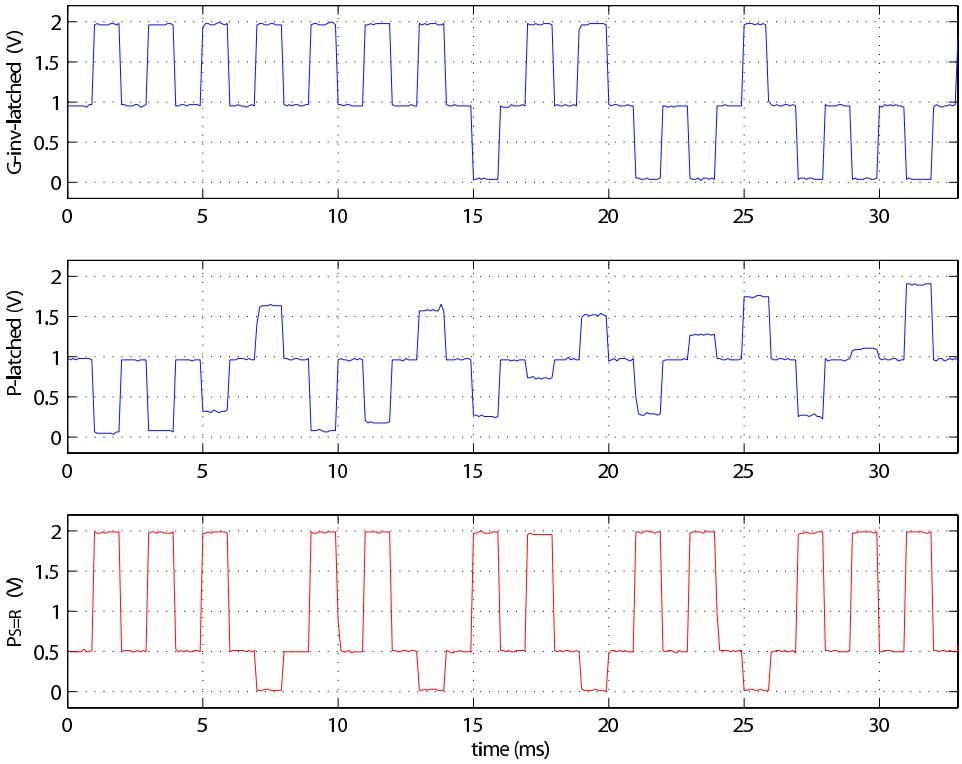

Figure 3.10: *Measurement of the NAND gate used to isolate the carry-sensitive sum of* **P**. $\overline{\mathbf{G}}$ *and* $\mathbf{P}_{\mathbf{latched}}$ *are used as inputs. The frequency is 500Hz.*

included. The reason for including the remaining gates, was to verify the design by measurements.

The gates of the Carry-Look-Ahead Generator included in the prototype MV CLA adder are two NOR gates and a binary inverter. As the carry-in needs to be included, whilst the generated carry must be added to the eventual propagating carry, NOR gates have been found to be ideal. The desired function of the gate summing **P** and the carry-in, is that only the propagating carry is represented by a logic **1** as output, as this output is added to the generated carry. To achieve a logic **1** for the desired values only, the carry-in needs to be inverted, as both the carry-sensitive value of **P** and the carry-in will be represented by a logic **0**. For this purpose a binary inverter is used. This inverter can seem redundant, since an inverted carry-in signal could have been used. The reason for choosing a positive carry-in and inverting it is explained in the section presenting the proposed CLA scheme. Figure 3.11 illustrates the NOR gates included in the prototype MV CLA adder as well as the inverter used to invert the carry-in signal.

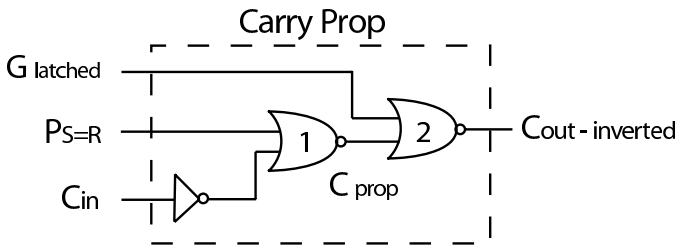

Figure 3.11: *The figure illustrates the binary NOR gates used in the Carry-Look-Ahead Generator, as well as the inverter used on the carry-in. The gates are included in the Carry Prop box, which is used in the later section presenting in the carry-look-ahead scheme. C9 = the unit capacitor C.*

Figure 3.12 and the corresponding truth table in Figure 3.13 verify the use of a NOR gate to realize the propagating carry.

The output of the first NOR gate represents the propagating carry (triggered by the carry-in), and along with the generated carry ($\mathbf{G_{latched}}$)

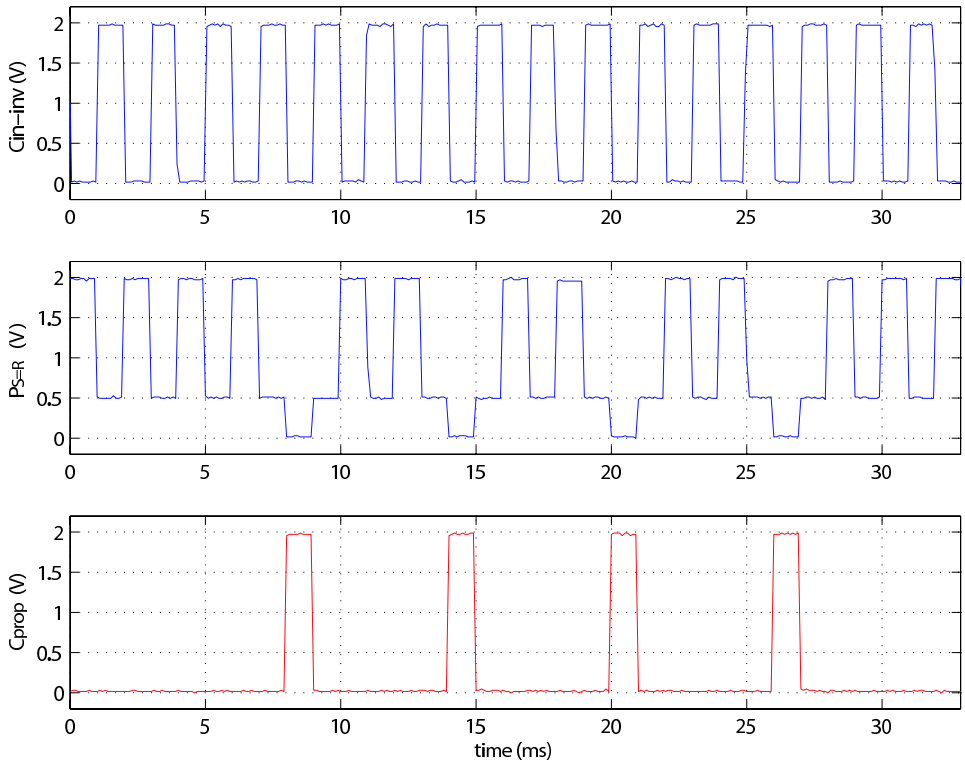

Figure 3.12: *Measurement of the NOR gate used to propagate a carry when* $\mathbf{P_{S=R}}$ *is* **0** *(sum = radix) and* $\mathbf{Carry_{in}}$ *is* **1**. *In this measurement the* $\mathbf{Carry_{in}}$ *is* **1**. *The frequency is 500Hz.*

C_{in}	$\overline{Z}$	$P_{S=R}$	$\overline{C_{in}}$	C_{prop}
1	0	1	0	0
1	1	1	0	0
1	2	1	0	0
1	3	0	0	1
1	4	1	0	0
1	5	1	0	0
1	6	1	0	0

Figure 3.13: *Truth table for the NOR gate used to determine the carry-propagation of a carry-in combined with* $\mathbf{P_{S=R}}$. *The carry-in sensitive value of* **P** *is marked in blue. For a better overview,* $\mathbf{C}_{in}$ *and the radix-8 sum* $\overline{\mathbf{Z}}$ *are also shown.*

the summation of these result in the carry out of the adder. To achieve a correct carry-out, the second NOR gate is used. As neither of these two carry signals give a carry simultaniously, due to the fact that any possible overlap has been removed in the earlier operations, a summation of the two signals through a NOR gate gives the desired result. It is worth noticing that the output, is an inverted carry-out, thus the name $\overline{\mathbf{C_{out}}}$. The reason for choosing an inverted carry-out is explained in the section presenting the proposed CLA scheme. Figure 3.14 showing measurements of the design, and the corresponding truth table in Figure 3.15 verify the design solution.

The adder element performing the actual addition of the input signals and the carry-in, is identical to the MV full-adder described in chapter two, Figure 2.12. This adder element is used to calculate the correct **Sum**, and is not used in the handling of the carry signal. The MV full-adder has been chosen for this purpose because of its simple and effective design. To be able to represent the **Sum** in phase with both **P** and **G**, a MV latch has been used. The MV latch used, is illustrated in the overview illustration of the prototype MV CLA adder. Figure 3.16 shows the schematic view of the prototype MV CLA adder designed for this thesis. Furthermore, a table of the transistor sizes used for the implementation is presented in Figure C.1. Measurements of the adder element, the latch included are pictured in Figure 3.17. The voltage level of the logic values of the latched **Sum** diverts from the corresponding simulated results in Figure 2.13. The measurements show that the radix-4 **Sum** is logically correct. Thereby the problem here lies in the latch used. As the

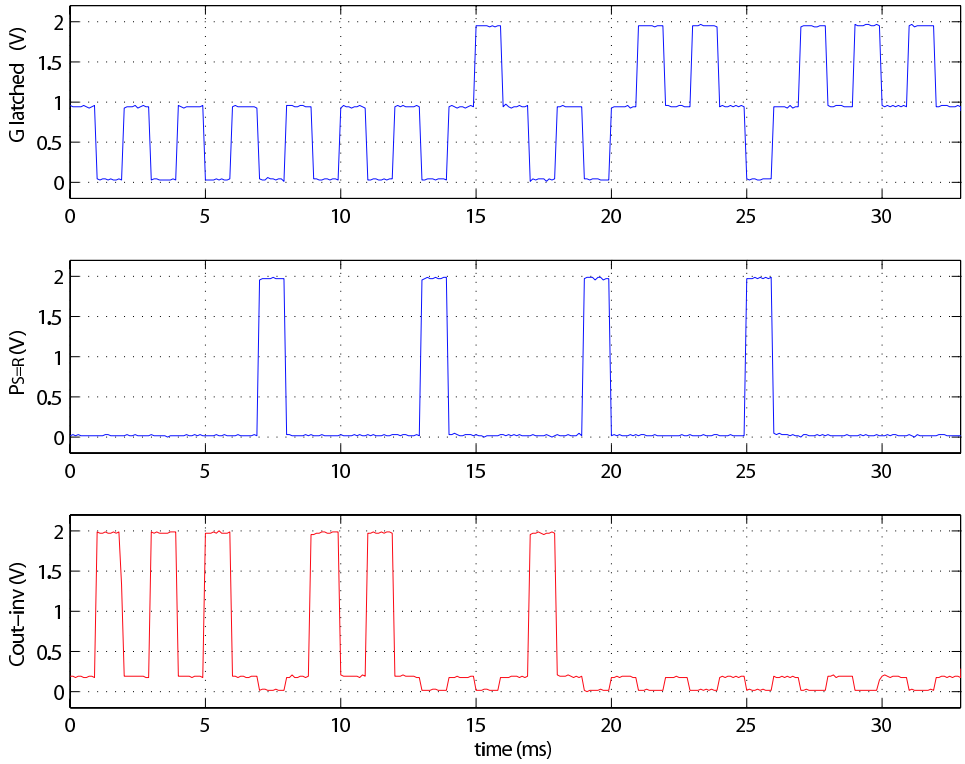

Figure 3.14: *Measurement of the NOR gate used to combine the propagating carry (*$\mathbf{C}_{PROP}$*) and the generated carry (***G***). The actual* $\mathbf{Carry}_{out}$ *is inverted, giving* $\overline{\mathbf{C}}_{\mathbf{out}}$*. The frequency is 500Hz.*

C_{in}	$\overline{Z}$	$G_{latched}$	C_{prop}	$\overline{C_{out}}$
1	0	0	0	1
1	1	0	0	1
1	2	0	0	1
1	3	0	1	0
1	4	1	0	0
1	5	1	0	0
1	6	1	0	0

Figure 3.15: *Truth table for the NOR gate used to determine the carry-out when the generated carry is combined with* $\mathbf{P}_{\mathbf{prop}}$*. For a better overview,* $\mathbf{C}_{in}$ *and the radix-8 sum* $\overline{\mathbf{Z}}$ *are also shown.*

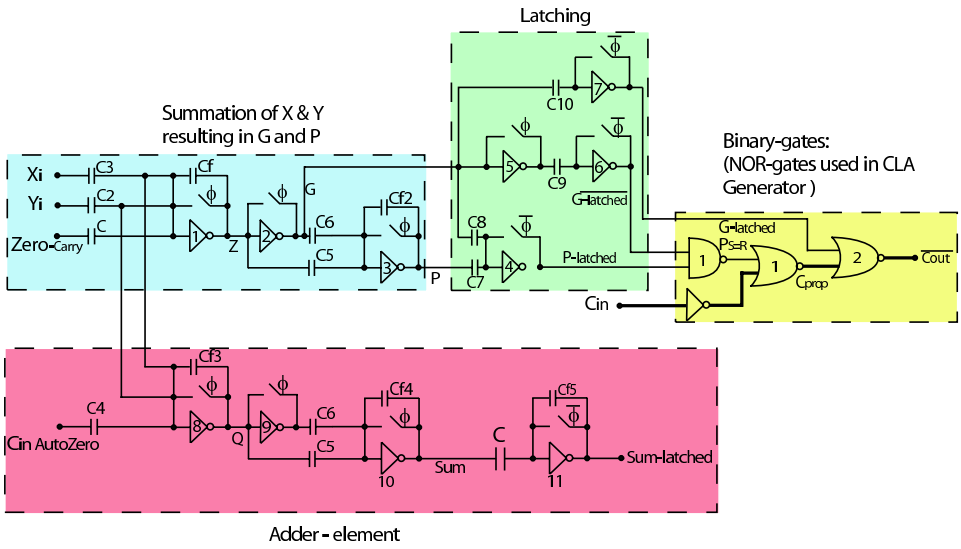

Figure 3.16: *The design shows the schematic view of the SFG MV Carry-Look-Ahead Adder The input is of ϕ and the output is of $\overline{\phi}$.*

focus of the thesis was not latches, this error has not been further addressed, although a better designed latch would have eliminated the error. However, the possible sources for these malfunctions are discussed in chapter four.

In Figure 3.16, the circuits described are combined, though divided in several sections for easier overview. The element performing the actual addition is marked with red color, while the elements performing the carry calculations are marked with light-blue, green and yellow.

A truth table illustrating the inputs (presented by the inverted node **Z**) and the carry-in along with $\overline{\mathbf{C_{Out}}}$ is illustrated in Figure 3.18. Truth tables describing the entire carry-handling element with the same pattern of inputs used for simulations and measurements are presented in Appendix A.

The motivation for binary carry calculation, whilst additions are multiple-valued, may need to be explained. One might wonder, since the carry calculation is binary, it would have been better to use a complete binary full-adder. The reason for choosing to implement the adder in multiple-valued logics, is that the addition is much more effective in MV, considering the complexity. Furthermore, carry computation is more effective in binary systems, than in multiple-valued logics. With this in mind, the prototype CLA adder has been designed as a hybrid solution.

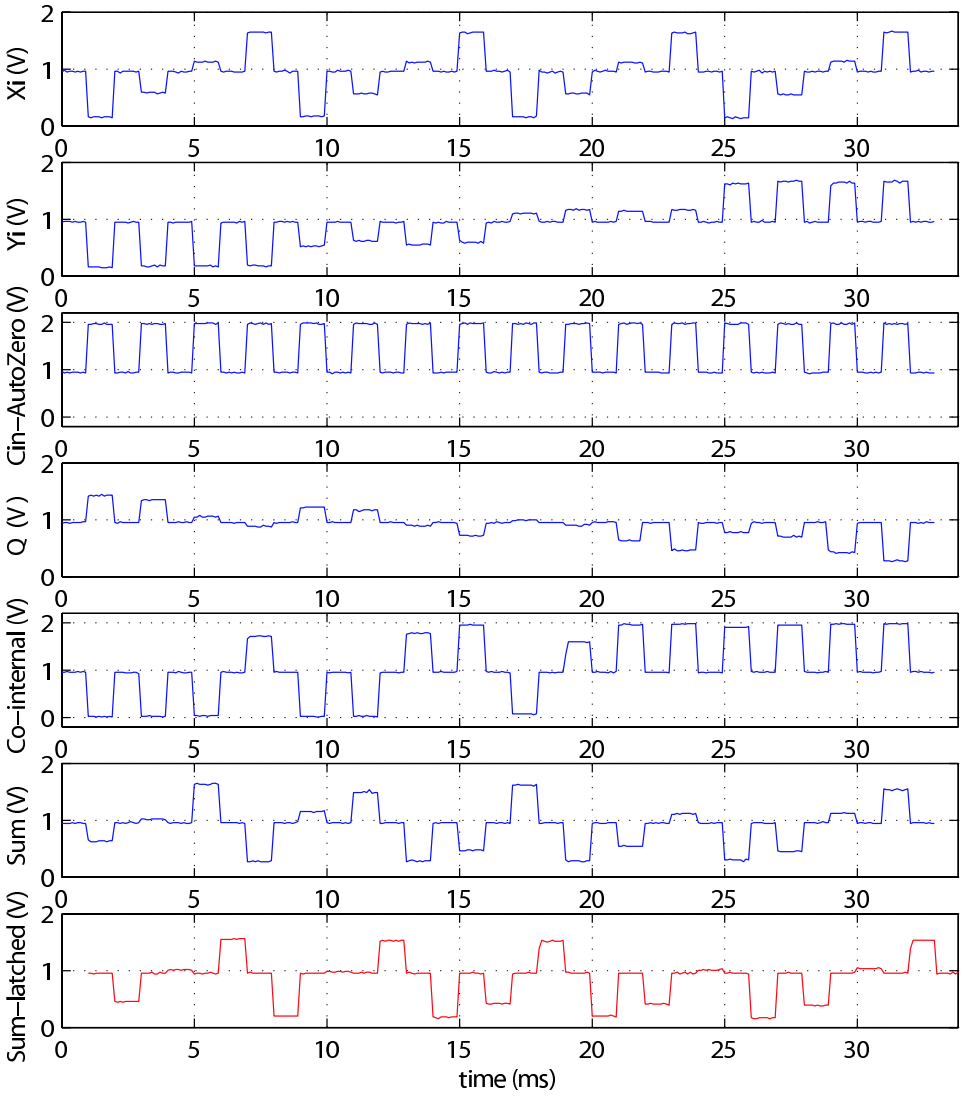

Figure 3.17: *Measurement showing the adder element of the circuit. The two R4 signals are generated by two BMVC's. The inverted Sum is represented by the R8 signal (node* **Q***). Further the Sum is latched to synchronize with the Carry-out of the circuit. The frequency is 500Hz.*

3.2 Prototype adder used in cascade

The presented CLA adder can be implemented using different schemes. To give a glimpse of the potential of the MV CLA adder, four MV CLA adders have been implemented in cascade, resulting in a 8-bit adder. In contrast with the carry-ripple of cascaded recharge adders presented in chapter two, the MV CLA adders offers the opportunity to include the

C_{in}	$\overline{Z}$	$\overline{C_{Out}}$
0	0	1
0	1	1
0	2	1
0	3	1
0	4	0
0	5	0
0	6	0

(a)

C_{in}	$\overline{Z}$	$\overline{C_{Out}}$
1	0	1
1	1	1
1	2	1
1	3	0
1	4	0
1	5	0
1	6	0

(b)

Figure 3.18: *Figure (a) illustrates the carry-out, with no carry-in, while figure (b) illustrates the carry-out with a carry-in. Node* $\overline{\mathbf{Z}}$ *represents the summation of the input signals* **X** *and* **Y**. *The numbers marked in blue represents the carry-in sensitive value of the sum.*

carry at a later stage, as seen in Figure 3.16. The implementation of four cascaded MV CLA adders is illustrated in Figure 3.19

The same peripheral components used for the single adder element, were used for the implementation of the four cascaded adders, in addition with buffers on the binary outputs and carry-out signals. The reason for adding buffers, is to ease the load caused by the pads used.

The cascaded 8-bit adder does not consist of identical adder elements. The reason for this is that the first adder element, whilst receiving a positive carry-in, sends out an inverted carry-out. This means that the second adder needs to handle an inverted carry. Therefore, the first adder element is exactly the same as the described prototype MV CLA adder. The remaining three adder elements on the other hand feature identical modifications. The carry-in used to calculate the **Sum**, needs to be latched to phase the input signals of the adders. Although the carry signals are latched in phase with the input signals, this does not affect the carry propagation. The carry propagation is handled by NOR gates. Furthermore, the inverter used to invert the carry-in for the carry calculation is not present in adder element two, three and four. Since the carry-out of all the adder elements are inverted, there is no need for the inverter in adder two, three and four. For this particular implementation, an inverted carry-in on the first carry-element would have been suitable, although this was not considered an option at the time. Furthermore this

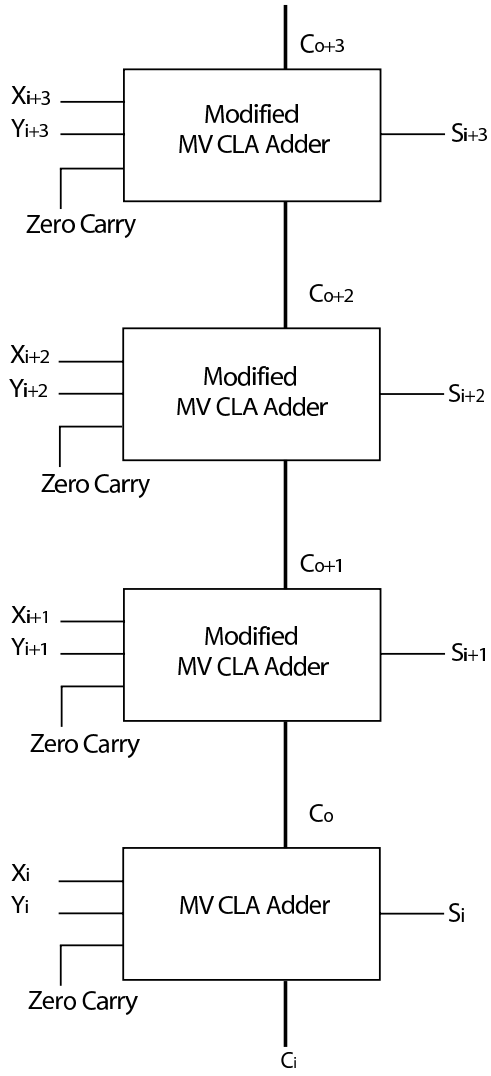

Figure 3.19: *The prototype MV CLA adder used in cascade.*

eight bit implementation was only designed to see that the carry signals ripple was correct from adder to adder. A simulation of the carry-ripple is illustrated in Figure 3.20.

As the carry-out is inverted, a logic **0** represents a carry propagation. The recharge signals from the MV calculations are also represented by logic **0**. The recharge-level of logic **0** has no effect on the calculation of the carry to the next element, as seen in the simulation. The simulation shows that a carry-in does not result in a shift of the signal. The shift comes if there is no carry-out, as the carry signal is low by default due to the recharge levels.

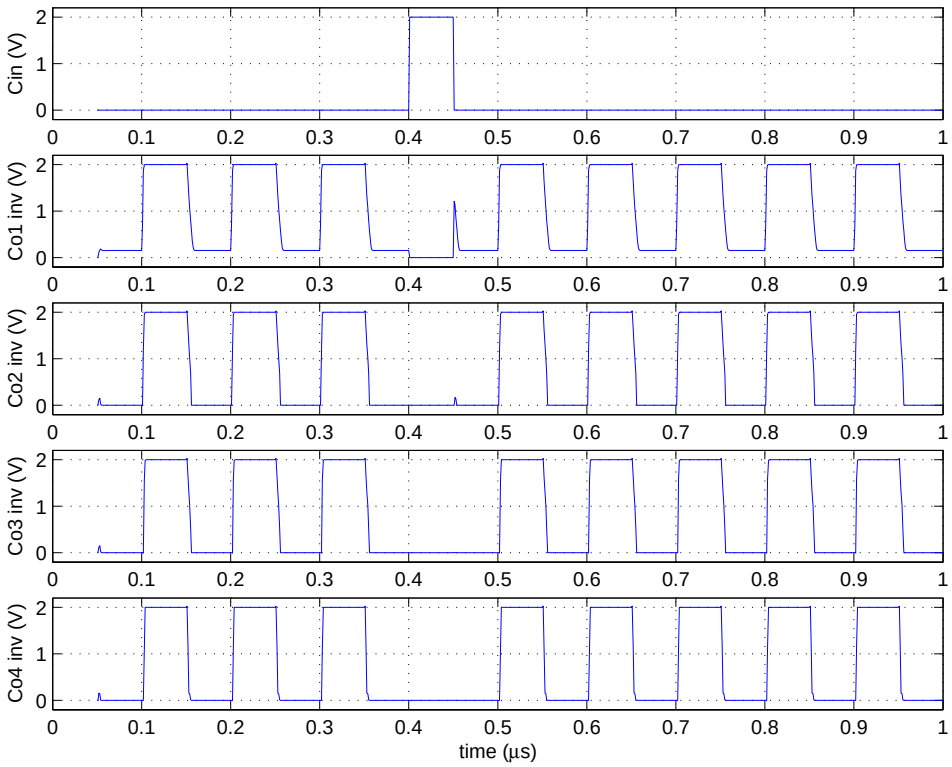

Figure 3.20: *Layout simulation of the cascaded prototype 8-bit MV CLA adder The simulation illustrates the carry-ripple through the four adder elements.*

Since the 8-bit cascaded MV CLA adder was only designed to verify the carry propagation, no gate delay of the carry propagation is presented. Furthermore, since the 8-bit cascaded MV CLA adder makes use of a carry-ripple scheme, it is not an actual carry-look-ahead adder. This means that it is far from optimized considering gate-delay on the carry signal. In the next section, a carry-look-ahead scheme featureing the MV CLA adder is presented.

3.3 Carry-Look-Ahead Scheme

In this section, the proposed carry-look-ahead scheme is presented. To obtain a better overview of the signal propagation, box-schematics are used. The 16-bit carry-look-ahead scheme presented is based upon the general example in Figure 1.1. For the carry-look-ahead scheme a modified version of the presented MV CLA adder is used. The modifications are made so that the generated carry and the carry-in sensitive sum, $\mathbf{G}_{\text{latched}}$ and $\mathbf{P}_{S=R}$ respectively, can be calculated in parallel before the

carry-in signal is introduced. The NOR gates and the inverter used for carry propagation, are represented by the box named “Carry Prop”. For the second, fourth, sixth and eight element, the inverter used for the carry-in has been removed since these adder elements need to handle an inverted carry-in. All adder elements, except the first, need to latch the carry-in used to calculate the **Sum** in order to phase the input signals. This on the other hand, does not affect the propagation of the carry signal.

The carry-in sensitive sums of two and two adder elements are combined, using a NOR gate. This is illustrated by the box named “S=R”. The carry-in is combined with $\mathbf{P}_{S=R}$ and $\mathbf{G}_{latched}$ in the boxes named “Next Level” by the use of NAND gates. “Next Level”also contains an inverter to invert the generated carry ($\mathbf{G}_{latched}$). The reason for inverting the generated carry is explained in the next section.

Figure 3.21 illustrates the proposed scheme. Notice that the inside-logics of the Carry-Look-Ahead Generator are represented in the figure, wires included. The red lines illustrate the parallel calculations. The boxes marked “next level” symbolize the logics where the carry-in is added. The idea of the carry-look-ahead scheme is that the carry-in is added at all levels simultaneously. Thus meaning that e.g. “next level 1” and “next level 4” calculate the respective carry-out simultaneously. This implies that e.g. “Adder 3” and “Adder 5” receive a carry-in simultaneously. The proposed carry-look-ahead is designed using a pyramid approach, from the lowest level, “next level 1”, to the top level, “next level 4”. It takes the same logic complexity to calculate the carry-out of “Adder 1”, as the carry-out of all the adders combined. The carry rippling lines ($\mathbf{C}_{ripple}$) between the elements is used only to calculate the correct sum, and never exceeds the depth of two adder elements.

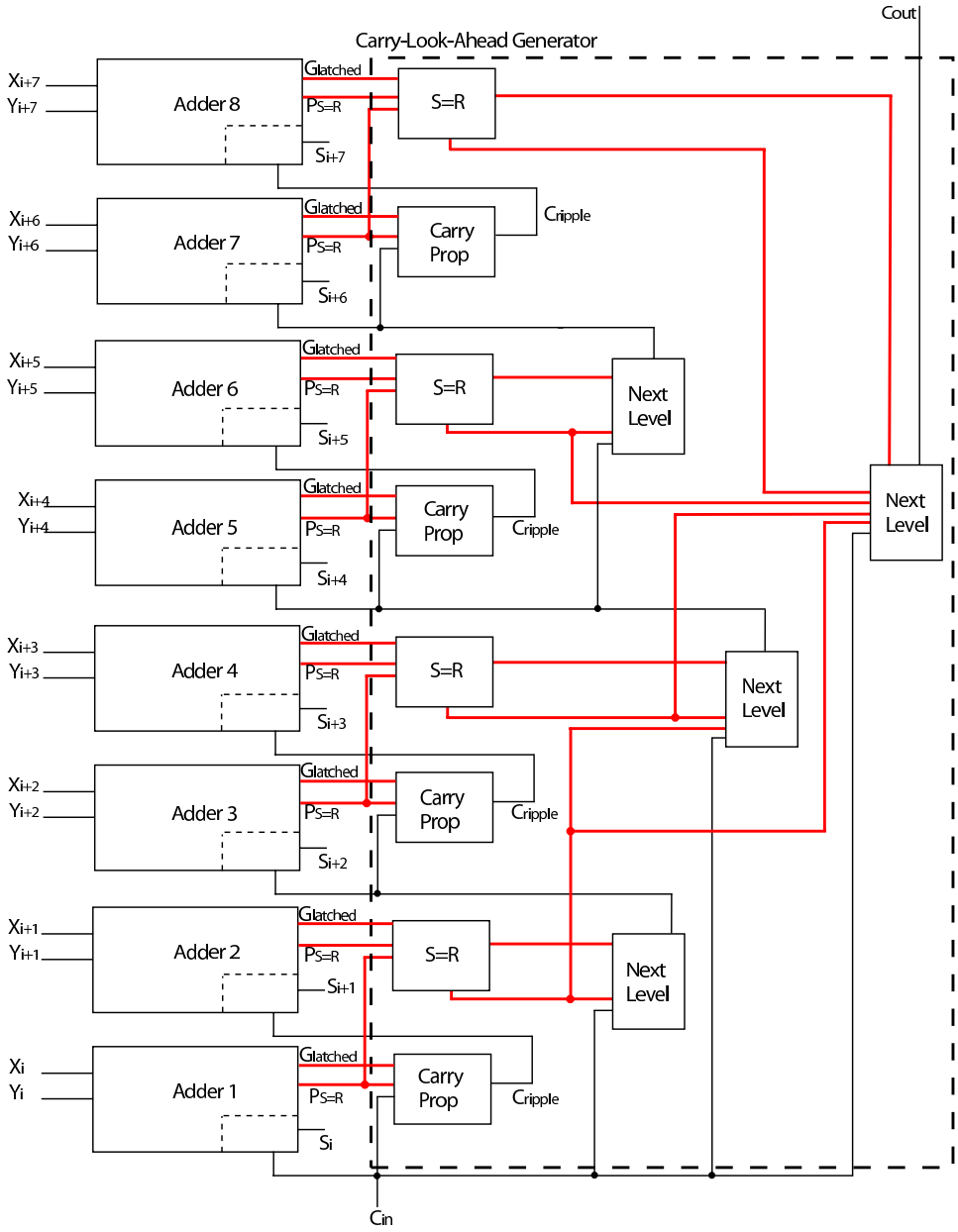

Figure 3.21: *The proposed expansion, using the CLA adder and additional logic gates. This illustration shows the expansion, using boxes for better overview.*

3.4 The MV CLA adder used in the CLA scheme

To fully understand the presented carry-look-ahead scheme, it is necessary to implement it with logic gates. Figure 3.22 shows the presented carry-look-ahead scheme fully integrated. As mentioned, modifications

have been made to the MV CLA adder to adjust it to the carry-look-ahead scheme.

The NOR gates summing $\mathbf{P_{S=R}}$ and $\mathbf{G_{latched}}$ have been moved outside of the adder and into the CLA Generator. Furthermore the carry-in sensitive signals from two and two adder elements are combined in a NOR gate. This can be illustrated using adder element one and two as an example. The output of this NOR gate is **1** only when both signals are carry-in sensitive. This signal is combined with the carry-in in a NAND gate (NAND 1). Here the reason for using a positive carry-in becomes evident. The output of the NAND gate is **0** only when the two combined $\mathbf{P_{S=R}}$ signals are carry-in sensitive and the carry-in is **1**. The output of this NAND gate is an inverted carry, thus the generated carry needs to be inverted before the signals can be summed using a second NAND gate (NAND 2). The output of the second NAND gate (Cout 2) is the carry-out of adder elements one and two. This example counts for the rest of the CLA generator as well. It is worth noticing that the carry-out of the CLA Generator (Cout 8) does not represent the worst path in respect of time-delay. The worst path is represented by the carry-in of the seventh adder element, as this carry signal has a gate-delay of $7\Delta_G$. This is still a large improvement compared to the worst path of the MV full-adder, of $16\Delta_G$. The carry-out of the CLA Generator only has a gate-delay of $2\Delta_G$, the same delay as for e.g. adder element 3 or 5. While the fan-in increases for each level of NAND gates used to calculate the propagating carry, the complexity of the levels does not increase. Each carry-look-ahead propagation level consists of only two NAND gates. The proposed CLA scheme is verified in Figure 3.23.

To further verify the increased efficiency presented by the proposed CLA scheme, gate-delay simulations are provided. The CLA scheme has been simulated on layout, using ideal wires. Furthermore to provide equal conditions, the transistor sizes used in MV logics are equal to the transistors used in the MV full-adder. The binary gates used in the CLA generator are designed using larger transistors, to be able to handle the larger binary voltage switching. Figure 3.24 illustrates the gate delay of positive carry-in. The spike on the “Worst delay” plot is recognized from Figure 3.23. The gate-delay of the carry-out of the CLA Generator is 1.0ns, while the gate-delay of the worst path is 1.7ns. These results imply that in terms of gate-delay, the result of the carry-out is **22** times better than the positive carry-out of the 16-bit MV full-adder. The gate-delay of the worst path settles the overall improvement to **13** times that of the 16-bit MV full-adder. These results are overall very good, but the negative flanks of the carry signals also need to be considered. Simulation illustrating this is provided in Figure 3.25. In this simulation,

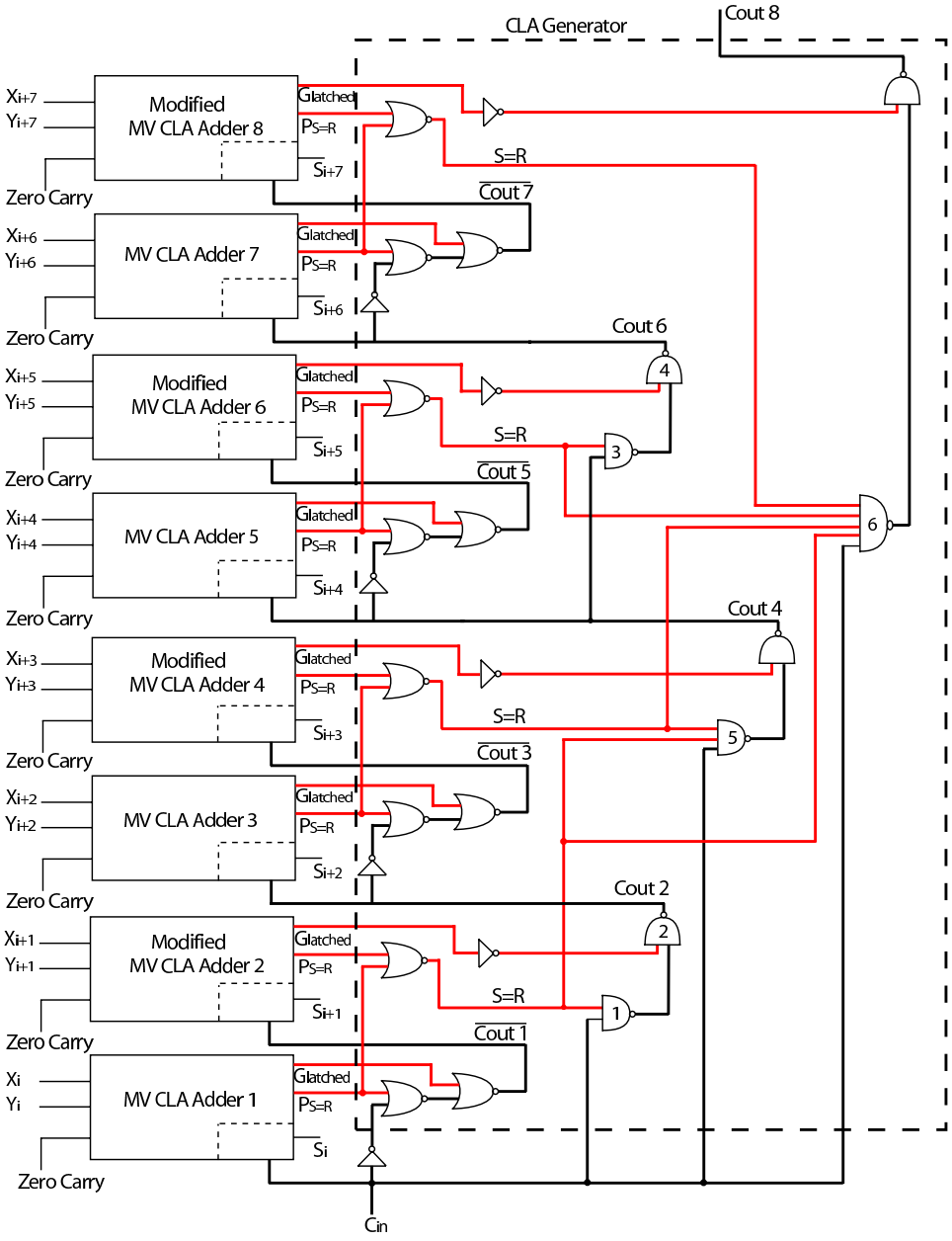

Figure 3.22: *The proposed expansion for optimized carry-propagation. The boxes are replaced by logics, and shows the suggested implementation. The red lines show the processes that happen in parallel, and are waiting for a Carry-in signal for further propagation.*

the gate-delay of the carry out of the CLA Generator is 0.7ns, while the gate-delay of the worst path is 2ns. Again, this implies that in terms of gate-delay, the carry-out is **57** times better than the carry-out of the 16-

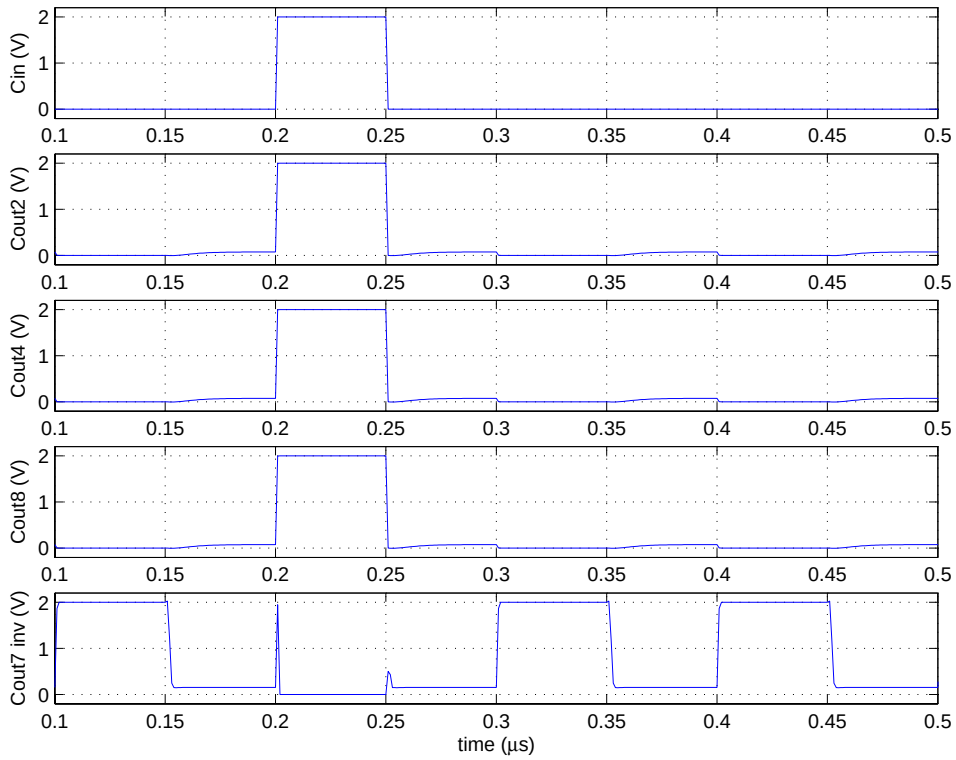

Figure 3.23: *Layout simulation of the CLA proposal, using ideal wires. The simulation illsustrates the scenario where* $\mathbf{G = 0}$, $\mathbf{P = 0}$ *(Sum = radix) and* $\mathbf{Carry_{in} = 1}$. *The frequency is 10MHz.*

bit MV full-adder. The worst path gate-delays for both adders are 40ns and 2ns for the 16-bit MV full-adder and the 16-bit MV CLA full-adder respectively. This implies that by using the proposed CLA scheme, an improvement of a factor twenty is achieved considering a carry-in triggered ripple. Furthermore, the worst delay of 2ns implies that the implemented CLA Generator can operate at frequencies up to 500MHz. Since the adder elements used are not affected by the gate-delay of the rippling carry signal, these too can operate at higher frequency than the adder elements of the 16-bit MV full-adder. The MV CLA adder has been simulated to operate logically correct at a frequency of 50MHz, illustrated in chapter four. This gives an overall frequency improvement of a factor **5**, when compared to the 16-bit MV full-adder.

The improvement in gate-delay also opens the opportunity to design larger adders. It is possible to expand the 16-bit MV CLA adder to a 32-bit MV CLA adder with only two additional NAND gates to handle the computation of a carry-in. The same counts for an expansion from

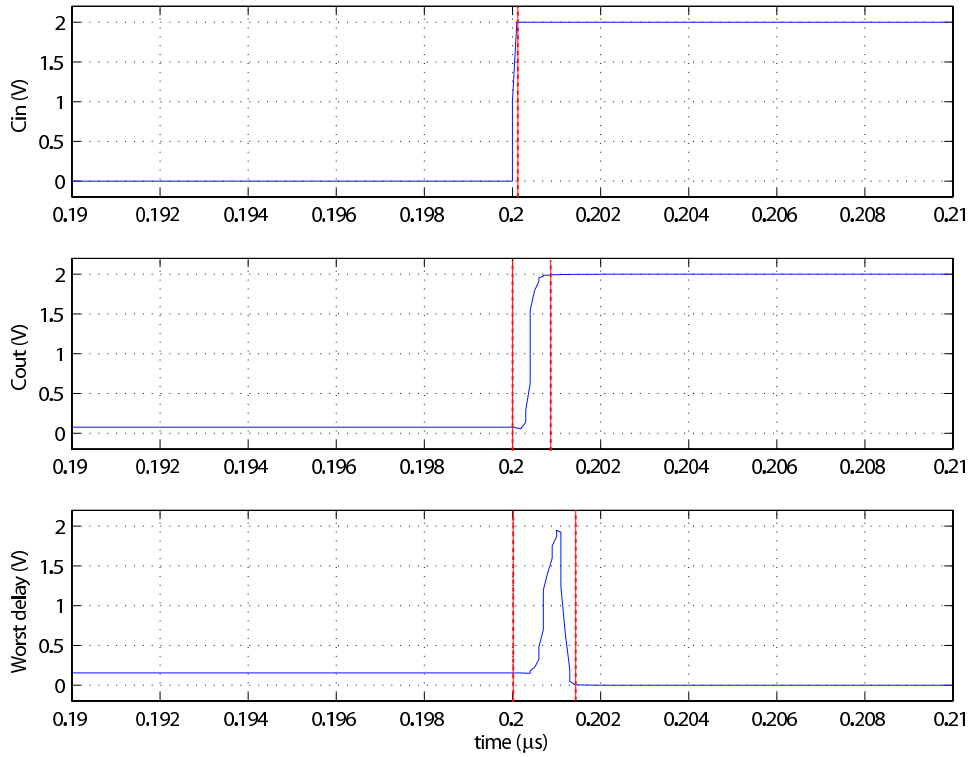

Figure 3.24: *Layout simulation of the CLA proposal, using ideal wires. The simulation illustrates the delay of the* $\mathbf{Carry_{out}}$ *of the 16-bit CLA adder, when both the carry-in and carry-out are* **1**. *Also shown, is the worst propagation line of the proposed CLA generator.*

32-bit to a 64-bit solution. Again, only two additional NAND gates are needed to ensure the logical depth of the carry propagation for a carry-in . Thus, with the proposed CLA scheme, large adders can be designed without the gate-delay of the carry-signal becoming critical. That said, the proposed CLA scheme is only one of many possible solutions.

Furthermore, it is worth noticing that the proposed CLA scheme does not include any logics to calculate the propagation of the generated carry-signals. There are several reasons for this, with the main reason being delay calculation. Since the generated carry **G**, is latched before it is introduced to the CLA Generator, the generated carry signals have been omitted when gate-delay calculations are made. Another reason for not including the logics used for generated carry propagation is accessibility. If the logics used to handle propagation of generated carry signals was included, it would be very difficult to obtain any overall overview of the system. A complete 16-bit MV CLA adder is therefore illustrated in Appendix B.

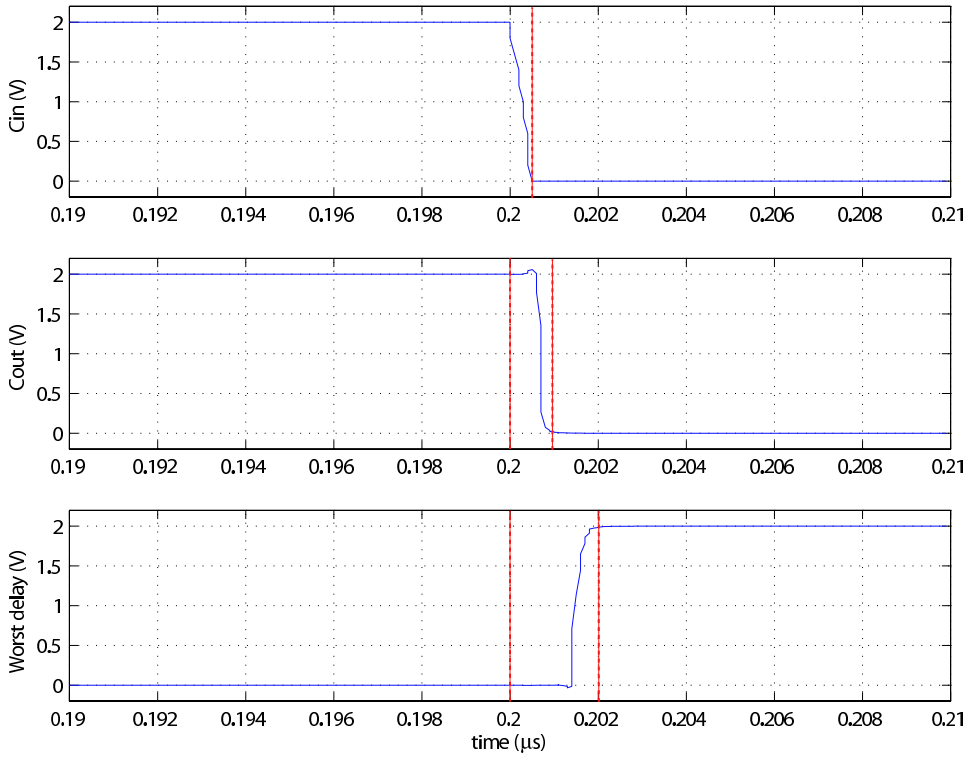

Figure 3.25: *Layout simulation of the CLA proposal, using ideal wires. The simulation illustrates the delay of the* $\mathbf{Carry_{out}}$ *of the 16-bit CLA adder, when both the carry-in and carry-out are* **0**. *Also shown, is the worst propagation line of the proposed CLA generator.*

3.5 Summary

The MV CLA adder presented offers the opportunity of a more efficient carry-handling than the recharge adder presented in chapter two, though at the cost of more area consumption. The cascaded version offers a smarter carry-ripple than the recharge adder, as the carry signal is binary and not as sensitive to gate delay. However, the presented MV CLA adder is also integrated in a proposed carry-look-ahead scheme. The proposed solution improves the performance considering both gate-delay and frequency, compared to the 16-bit MV full-adder. Therefore, the proposed solution can increase performance of multipliers and decimators, regarding both number of bits and frequency. The fabricated circuits have been measured and verified to work satisfactory, although some results appear to include some form of mismatch. In the next chapter possible reasons for mismatch are discussed, with solutions also described.

Chapter 4

System Considerations

Although the presented circuits work perfectly in theory, and also when simulated in layout, mismatch in the components can alter the functionality of the system. In this chapter possible sources for malfunction are discussed, with possible solutions also presented. Furthermore power consumption and max frequency of the MV CLA adder are discussed. The layout of the prototype design is also described, providing argumentation for the design methods used.

4.1 Sources of malfunction

When implementing multiple-valued designs, as with any other design technique, there are always elements that can cause malfunction. Potential sources of malfunction are discussed in this section. To obtain the most linear representation of the MV signals, the operation range of the signals has been set between 0.2v - 1.8V. The reason for this is that the gain decreases near V_{dd} and V_{ss}, since the transistors enter the linear region. Although helpful, this precaution is not enough to prevent mismatch as other factors contribute as well.

First the problem of mismatch between the transistors is considered. A mismatch between the nMOS and pMOS transistor in a recharge inverter, will result in a recharge level different of $V_{dd}/2$. Furthermore, a mismatch between the recharge transistors, will contribute to a mismatch to the voltage swing (ΔV) of the logic levels. Figure 4.1 shows a simulation of the worst possible scenario on a single chip, considering nMOS and pMOS matching. The worst-one (Wo, weak pMOS in Cadence) simulation and the worst-zero (Wz, weak nMOS in Cadence) simulations are compared to the typical mean (Tm) simulation.

While layout simulations (Tm) of the designed system suggested that the recharge inverters were matched, measurements suggested otherwise, as seen in Figure 4.2. The measured result shows great resemblance with the Wo simulation made, suggesting that the prototype chip has weaker nMOS transistors than desired. Furthermore it is worth noticing that like the Wo simulation, the measured recharge signal has a recharge level below $V_{dd}/2$, more precisely around 0.956V. It is therefore fair to state that the nMOS transistors are stronger than the pMOS transistors on the prototype chip. Several methods to reduce transistor mismatch have been proposed [27–29].

The measured signal in Figure 4.2 is the output of the BMVC, and one of the input signals used in the designed adder. Even with signals that are deviant to the desired input signals considering ΔV the adder operates satisfactory. It is therefore fair to state that the presented design is robust.

In addition to transistor mismatch, another factor that contributes to a mismatch of the voltage change (ΔV), is mismatch in the capacitors.

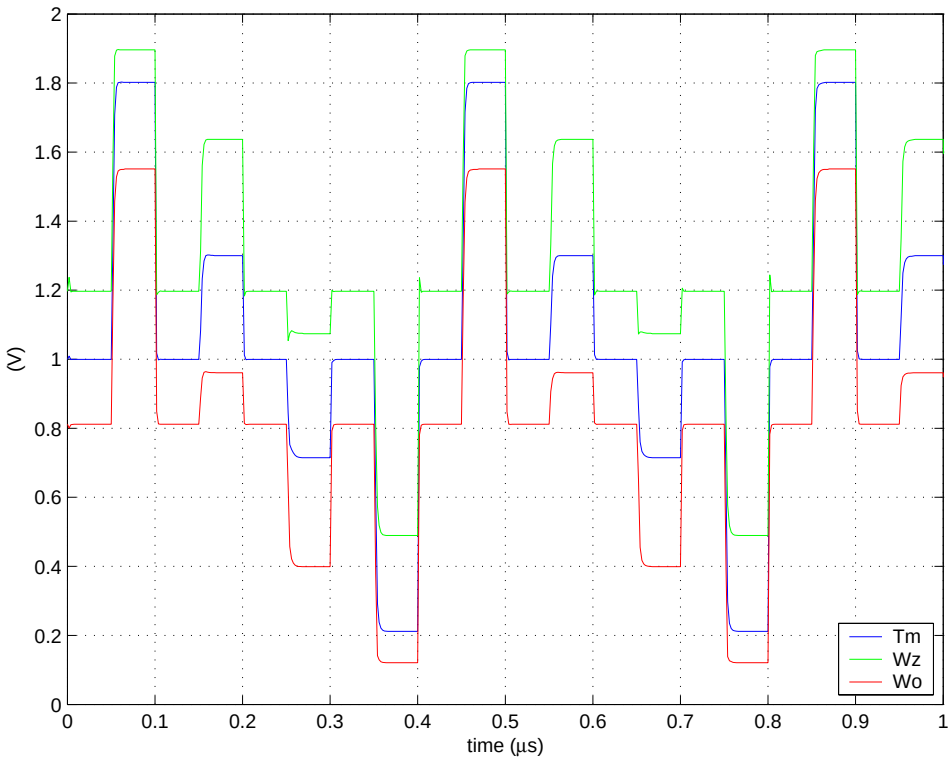

Figure 4.1: *The simulation illustrates the differences of Tm, Wz and Wo simulations.*

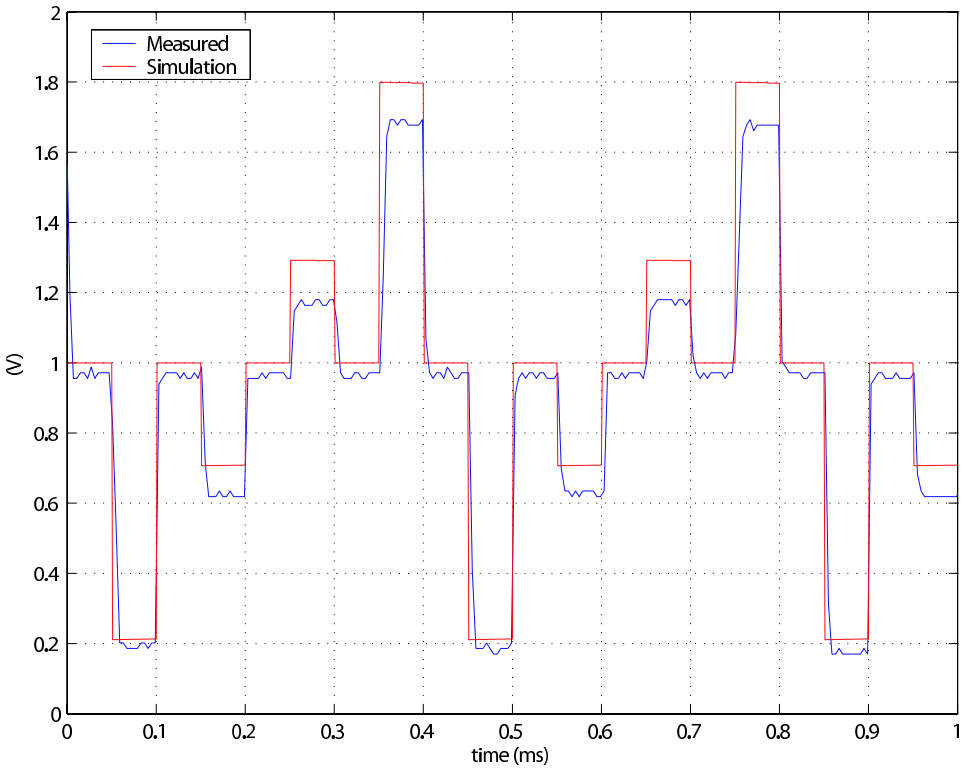

Figure 4.2: *The figure shows the difference between simulation, with all parasitic capacitances added, and measure of the Binary to Multiple-Valued Converter.*

The mismatch can occur between the input capacitors, or even at the feedback capacitor. A mismatch between the input capacitors will result in a mismatch of the weighing of the signals, whilst a mismatch in the feedback capacitor will result in a smaller or larger gain on the output as seen in equation 2.1. A smaller gain will decrease the ΔV of the output, whilst a larger gain will increase the ΔV of the output. Both cases contribute to irregularities concerning the linearity of the output signal. Further considerations regarding capacitors are mentioned in the section discussing the prototype design.

Another factor that has to be taken into consideration is that the unit capacitor **C** used in the BMVC, is 5.09fF. For a constant offset due to process variations, the relative deviation in capacitance will be larger for a minimum sized capacitor than for a capacitor twice the size, increasing the sensitivity to such variations.

The above-mentioned irregularities contribute to the fact that adder elements representing more than 2-bits, become very sensitive to error margins and parasitic capacitances.

4.2 Clock Frequency

In this thesis the frequencies stated, are the clock frequency. Therefore the frequencies of the recharge signals contain both the recharge period and the evaluation period. As the measurements on the prototype chip were made to verify the functionality of the design, simulations are relied on to determine max frequency of the design. The implemented MV CLA adder has been simulated to run flawlessly at 50MHz in layout with all capacitances included, This was verified using equal transistor sizes as the MV full-adder for the MV inverters. At this frequency the logic values, though correct tend to be closer to $V_{dd}/2$ than desired. Since the CLA Generator can handle frequencies up to 500MHz, implies that one can extend the proposed 16-bit MV CLA adder scheme with a significant number of bits before carry handling becomes problematic. The 50MHz simulation is illustrated and verified in Figure 4.3.

It is important to stress the fact that the prototype chip was not designed to handle high frequencies. The purpose was and is to see how the concept works in an actual implementation, an how well the different logic levels can be distinguished. Therefore, a MV design able to handle higher frequencies may be desired.

For this thesis a single phased clocking scheme, of ϕ and $\overline{\phi}$ has been used. The disadvantages of such a clocking scheme are obvious. The fact that the clock skew can decrease the gain, was considered a risk worth taking as the simplicity of such a clocking scheme is desirable for a prototype design using low frequencies. To reduce the power consumption caused by the clocks, other more sophisticated clocking schemes can be used [30].

The test circuit has no buffers on the probe-points, as they are multiple-valued signals. Considering the pads used for the chip have a load of 5pF, the prototype 2-bit adder has been tested with a clock frequency of 500Hz.

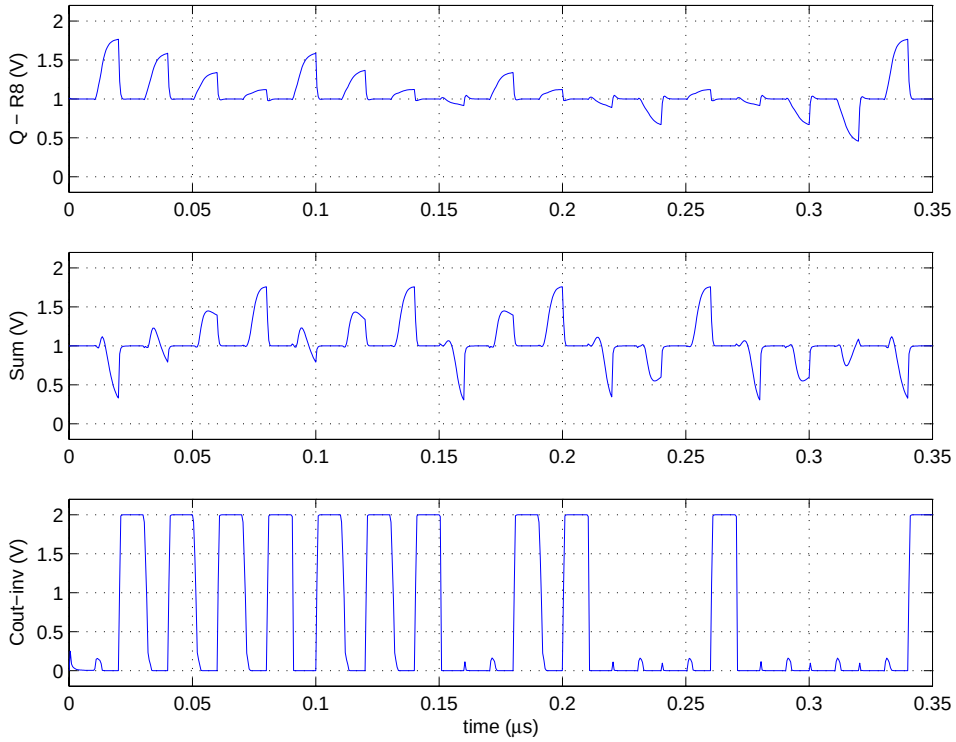

Figure 4.3: *Layout simulation of the MV CLA adder. The frequency is 50MHz.* **Q** *(radix-8 sum of* **X**, **Y** *and* $\mathbf{C_{in}}$*), the radix-4* **Sum** *and* $\overline{\mathbf{C_{out}}}$ *are shown.*

4.3 Power consumption

While MV circuits have a higher average power consumption than binary circuits, the power consumption of MV circuits is also far more static than for binary circuits. Whilst binary circuits have to be designed to handle large voltage swings on the gate, MV circuits experience only small voltage changes, due to the floating-gate. The power consumption of the binary and the MV circuits used for the prototype chip is illustrated in Figure 4.4. In order to provide the most extreme result, the worst case transition (Worst power Wp) has been used. Matlab calculations show that the average power consumption for the MV circuits used, is $\mathbf{5.5127^{-4}}$A. The binary circuits on the other hand, have an average power consumption of $\mathbf{4.0215^{-4}}$A. To achieve a better understanding of the power consumption of the circuits discussed, the peak values of both were also calculated in Matlab. While the MV circuits show a peak of $\mathbf{6.7126^{-4}}$A, the binary circuits have a peak of $\mathbf{9.2939^{-4}}$A. This implies that the MV circuits have a peak value that is a factor **1.2** of the average power consumption. The binary circuits have peak value that is a factor **2.3** of the average power consumption. Furthermore a comparison of

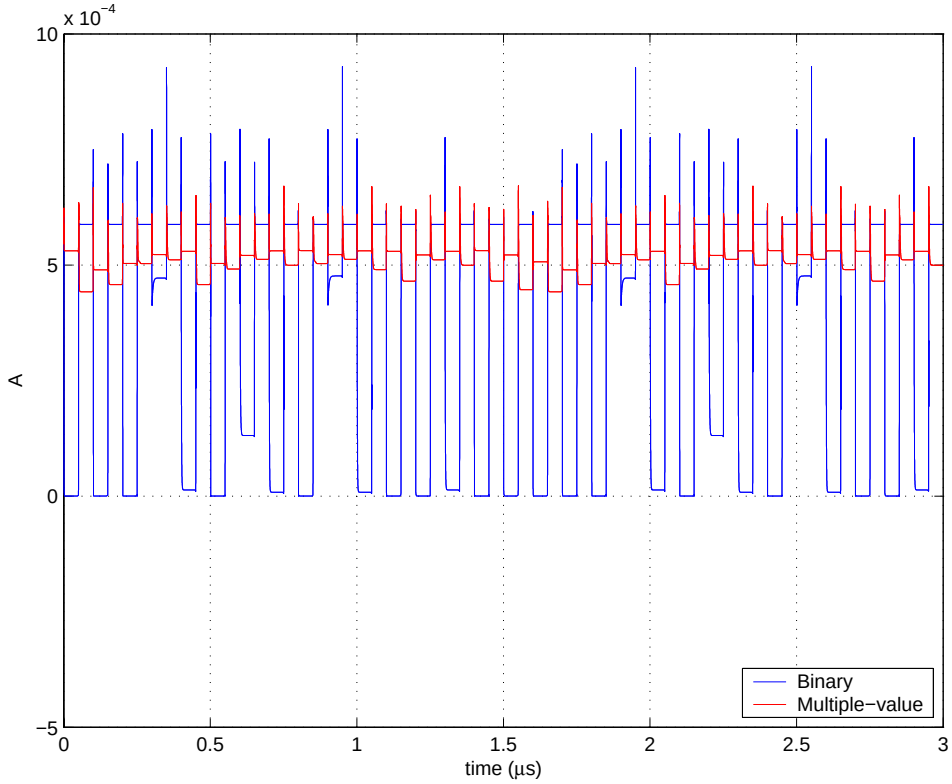

Figure 4.4: *The simulation illustrates the power consumption of the recharge elements and the binary gates used in the MV CLA adder.*

the peak amplitude values of the two different logics has been made. While the binary circuits used have a peak amplitude of $\mathbf{9.2938^{-4}}$A, the MV circuits have a peak amplitude of $\mathbf{2.29^{-4}}$A. This comparison further illustrates the static power consumption of MV circuits. The calculations made in Matlab, imply that binary circuits are best used for operations that are critical considering speed, like carry computation.

To further illustrate the static effect of the power consumption of MV circuits, a **log** plot is provided in Figure 4.5. The logarithmic power consumption of the binary circuits is also shown. From the figure it is evident that the MV circuits have a static power consumption.

The 16-bit MV full-adder and the 16-bit MV CLA adder have been compared by terms of gate-delay and max frequency, but another aspect worth comparing is power consumption To obtain the most correct result, both adders have been simulated at the same frequency (10MHz), with equal $V_{dd}$of 2V. While the 16-bit MV CLA adder is better than the 16-bit MV full-adder at both max frequency and gate-delay, it has a higher

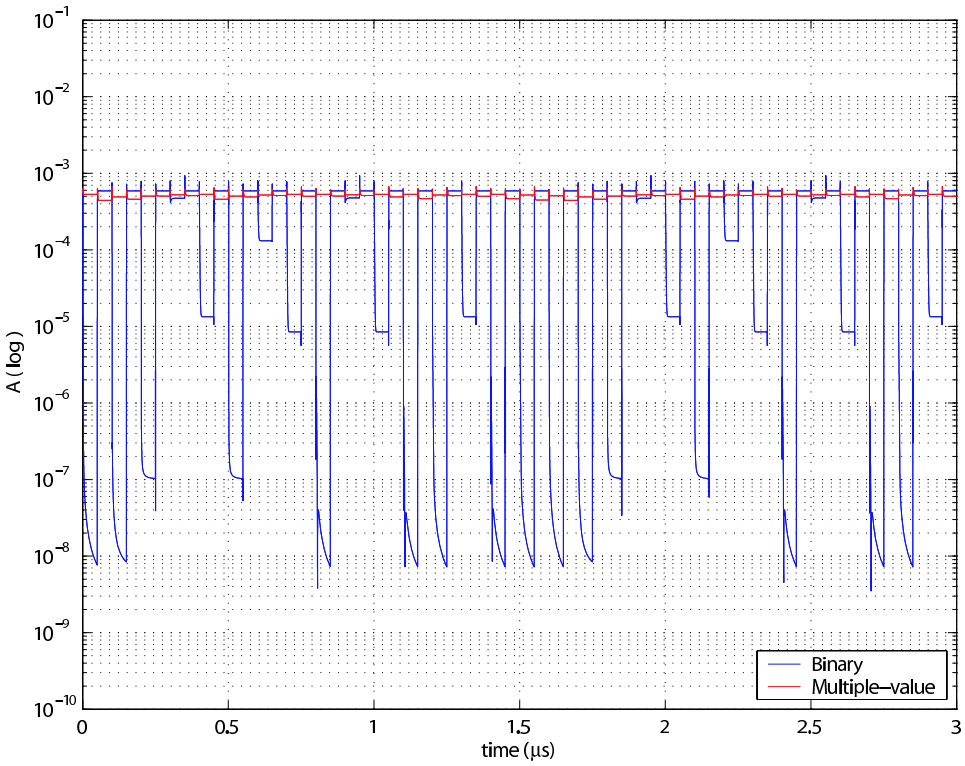

Figure 4.5: *Log plot of the power consumption for the recharge elements and the binary gates used in the MV CLA adder.*

power consumption than the 16-bit MV full-adder. While the 16-bit MV full-adder has a power consumption of $\mathbf{3.404^{-3}}$A, the power consumption of the 16-bit MV CLA adder is $\mathbf{7.493^{-3}}$A. Power alone is a questionable metric, since it can be reduced simply by computing more slowly. To obtain a more correct presentation of the power consumption, the power-delay product (PDP) and the energy-delay product is presented. The PDP and EDP values of both adders are presented in Figure 4.6. Both PDP and EDP calculations show that the 16-bit MV CLA adder has a lower energy consumption than the 16-bit MV full-adder. The PDP and EDP calculations show that the 16-bit MV CLA adder is more energy effective than the 16-bit MV full-adder. EDP calculations provide the most accurate result, since the EDP value is less sensitive to V_{dd} and frequency changes. The difference in EDP values of the two designs is significant. While the 16-bit MV full-adder has a EDP value of $\mathbf{5.447^{-18}}$J·sec, the EDP value of the 16-bit MV CLA adder is $\mathbf{2.997^{-20}}$J·sec.

	PDP	EDP
16-bit MV Full Adder	1.362^{-10}J	5.447^{-18}J·sec
16-bit MV CLA Full Adder	1.499^{-11}J	2.997^{-20}J·sec

Figure 4.6: *The table illustrates the PDP and EDP values of the 16-bit MV full-adder and the 16-bit MV CLA full-adder.*

4.4 The prototype design

Each of the components used making the prototype chip, were implemented using the Austria Microsystems (AMS) 0.35 μm process. All of the components are tested in schematic both as single components, and as parts of the overall system. The values of the capacitors are calculated considering both signal-weight and whether or not the addressed signal uses ΔV of 0 - 2 V or ΔV 0.2 - 1.8 V.

As for the layout of the designed chip, all multiple-valued circuits have been designed with the same method as illustrated in Figure 4.7. With the transition from schematic to layout, the capacitors needed adjustment due to the parasitic capacitances added. As with the schematic design, the layout of the chip was thoroughly tested as a system.

The capacitors of the prototype chip were kept as simple as possible. With the layout design presented, simple capacitor designs were desirable, as these are easier to implement. If one decided to use matched capacitances, dummy-capacitances would be needed as well [31]. Again, this would complicate the design, and increase the area consumption dramatically. As the dummy-capacitances need to be connected to V_{ss}, the total capacitance seen from node **X** in Figure 4.8 would increase, compared with the use of different sized capacitances. The fact that the dummy capacitors are connected to V_{ss} , implies that using such capacitors would help minimize charge-injection. However, the gain would also be reduced, and thereby decrease the error margin of the signal. This was also the case in layout simulations. Therefore, the prototype design was made without dummy capacitors. Also, guardrings were placed around the capacitors, to protect them from noise.

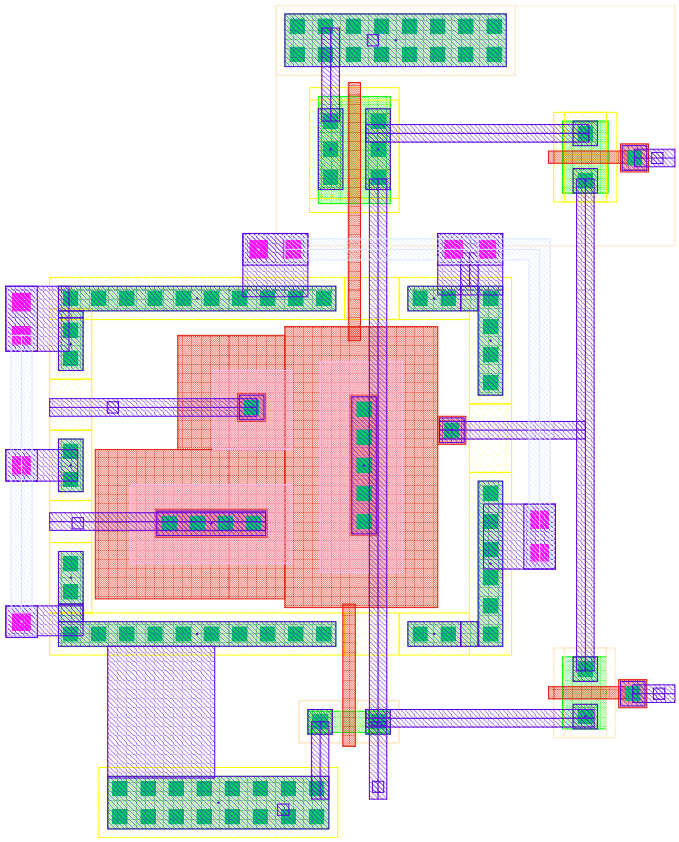

Figure 4.7: *A typical implementation used for R_m inverters. In stead of making three separate capacitors the capacitors were designed using a single base of $poly1$ and three pieces of $poly2$. Guardrings are placed around the inverters, to protect them from noise.*

Another design solution worth mentioning, is the routing of the clock-signals. To reduce the noise created when the clock signals shift, the wires of ϕ and $\overline{\phi}$ have been laid on top of each other using two layers of metal.

Area consumption was not a critical point for the design of the prototype chip. The design was implemented with focus on easy access for the probe points used. The prototype MV CLA adder is illustrated in Figure 4.9, while the 8-bit MV CLA adder is presented in Figure 4.10.

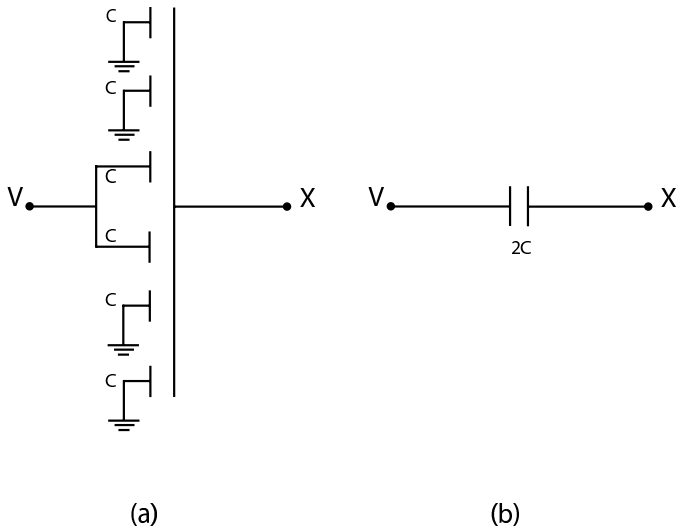

Figure 4.8: *Figure a shows the total capacitance seen from node X, using matched capacitances and Dummy-capacitances. The solution chosen can be seen in figure b. Notice that the total capacitance seen from node X is much larger in Figure a.*

4.5 Summary

In this chapter possible sources of malfunction have been discussed. Furthermore, solutions that help prevent these malfunctions have been presented. From the discussion conclusions can be made that recharge logics have lower variation in power dissipation, than binary circuits. Furthermore, the 16-bit MV CLA adder has been calculated to be more power effective than the 16-bit MV full-adder. The design choices considering the prototype design have been discussed.

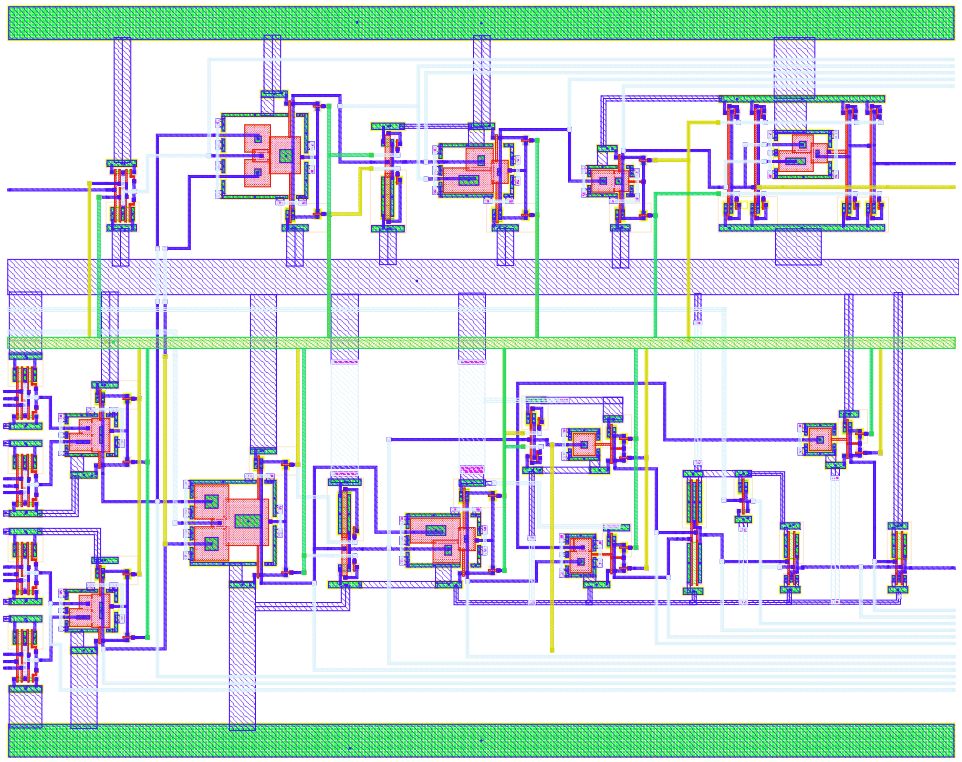

Figure 4.9: *The designed prototype MV CLA full-adder. Also shown, are the probe points used for measurements.*

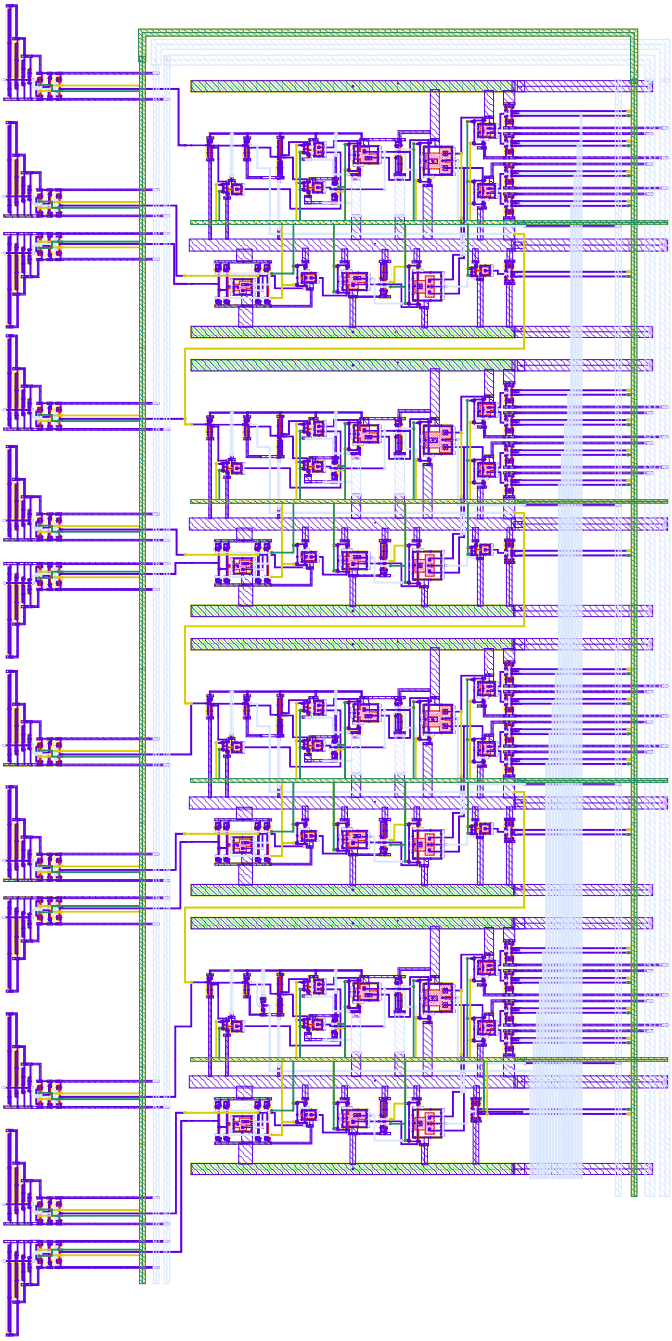

Figure 4.10: *The designed prototype 8-bit MV CLA full-adder.*

Chapter 5

Conclusion and proposal for further work

In this thesis a voltage-mode multiple-valued recharge logic carry-look-ahead full-adder has been presented. The MV CLA full-adder includes the use of floating-gate, or more precisely semi-floating-gate, as well as binary gates. The operation of the semi-floating-gate is determined by the weighed summation of voltage on the multiple input signals.

The essence of this thesis lies in the multiple-valued recharge carry-look-ahead scheme presented, as well as implementation of the prototype Carry-Look-Ahead full-adder presented. Since the generated carry signals are latched, these are not considered in the the gate-delay calculations. Therefore the focus has been on the carry propagation generated by a carry-in. A full implementation of the CLA scheme is presented in Appendix B. The circuits were fabricated in a CMOS 0.35μm process in order to measure them on an actual ASIC.

5.1 Main contributions

The discoveries and ideas of the author of this thesis are presented in chapter three, with the proposed carry-look-ahead scheme as main contribution. The proposed scheme offers logic depth achieved with few extra components, considering the propagating carry triggered by a carry-in.

5.2 Experimental Results

Each circuit was simulated as a single element and as a part of an overall system, and were proven to operate satisfyingly. Measurements show

that there is a mismatch in the circuits, resulting in a recharge level of 0.956V, as well as a mismatch in the voltage swing (Δ_V) of the logic values. Although a mismatch in the system was discovered, the adder works satisfactory, with the exception of a single bit error on the output of the MVBC used to convert the Sum. This bit error is caused by the latch, used with the radix-4 Sum. The actual Sum is logically correct. Due to the MV circuits being sensitive to error margins and parasitic capacitance, adder elements larger than 2-bit are not recomended.

The implementation of the presented 16-bit CLA scheme has showed to have smaller gate-delay than the implementation of the 16-bit MV full-adder. It has been verified that the implemented MV CLA adder also can operate on higher frequencies than the implemented MV full-adder, with both adders designed with equal sized transistors. Furthermore, Matlab calculations show that the implemented MV CLA full-adder has lower PDP and EDP values than the implemented MV full-adder. This implies that the proposed MV CLA full-adder is more energy effective than the MV full-adder. It is therefore fair to state that a carry-look-ahead scheme can increase the performance of multiple-valued adders. Applying the proposed CLA scheme can therefore also lead to an increase in performance for systems using MV SFG adders. While the implemented MV CLA adder has been simulated to operate at 50MHz, the implemented CLA Generator is estimated to operate at frequencies up to 500MHz. It is therefore desireable to design MV adder elements capable to operate at higher frequencies than 50MHz, and thus exploit the potential of the CLA scheme better.

5.3 Further work

As the designed prototype adder has been proven to work, implementing it with the proposed carry-look-ahead scheme on a fabricated chip would be worth exploring. Also implementing the proposed carry-look-ahead scheme in other applications that make use of the adders, e.g. a multiplier. Furthermore the proposed carry-look-ahead scheme could be implemented in a decimator, also here providing a more effective carry handling. Last but not least, publications could be made upon the discoveries made.

Appendix A

Truth tables

The truth tables presented here illustrate the input signal combinations (represented by node **Z**) used for testing the prototype adder. The first table illustrates the carry calculations with carry-in of **0**, and the second table illustrates the carry calculation when carry-in is **1**.

Z	**G**	**P**	P_{latch}	$\overline{G_{latch}}$	$P_{S=R}$	$\overline{C_{In}}$	C_{prop}	G_{latch}	$\overline{C_{out}}$
0	0	0	0	1	1	1	0	0	1
1	0	1	0	1	1	1	0	0	1
2	0	2	0	1	1	1	0	0	1
3	**0**	**3**	1	1	**0**	1	0	0	1
1	0	1	0	1	1	1	0	0	1
2	0	2	0	1	1	1	0	0	1
3	**0**	**3**	1	1	**0**	1	0	0	1
4	1	0	0	0	1	1	0	1	0
2	0	2	0	1	1	1	0	0	1
3	**0**	**3**	1	1	**0**	1	0	0	1
4	1	0	0	0	1	1	0	1	0
5	1	1	1	0	1	1	0	1	0
3	**0**	**3**	1	1	**0**	1	0	0	1
4	1	0	0	0	1	1	0	1	0
5	1	1	1	0	1	1	0	1	0
6	1	2	1	0	1	1	0	1	0

Table A.1: *The propagation of the carry-signal through the Carry Element, with a Carry-in signal = **0** is illustrated. The essential Bits described in the text are bold.*

Z	**G**	**P**	P_{latch}	$\overline{G_{latch}}$	$P_{S=R}$	$\overline{C_{In}}$	C_{prop}	G_{latch}	$\overline{C_{out}}$
0	0	0	0	1	1	0	0	0	1
1	0	1	0	1	1	0	0	0	1
2	0	2	0	1	1	0	0	0	1
3	**0**	**3**	1	1	**0**	0	1	0	0
1	0	1	0	1	1	0	0	0	1
2	0	2	0	1	1	0	0	0	1
3	**0**	**3**	1	1	**0**	0	1	0	0
4	1	0	0	0	1	0	0	1	0
2	0	2	0	1	1	0	0	0	1
3	**0**	**3**	1	1	**0**	0	1	0	0
4	1	0	0	0	1	0	0	1	0
5	1	1	1	0	1	0	0	1	0
3	**0**	**3**	1	1	**0**	0	1	0	0
4	1	0	0	0	1	0	0	1	0
5	1	1	1	0	1	0	0	1	0
6	1	2	1	0	1	0	0	1	0

Table A.2: *The propagation of the carry-signal through the Carry Element is illustrated. This table shows the result of a Carry-In =* **1**. *The essential Bits described in the text are bold.*

Appendix B

Complete CLA proposal

B.1 A complete 16-bit MV CLA full-adder

A complete proposal of a 16-bit MV CLA full-adder is presented in Figure B.1. This implementation has not been designed in either schematic or layout, due to time constraints. Although a presentation of the gates used to calculate the propagation of the generated carry signals is provided.

The first NAND gate (NAND 1) Sums $\mathbf{G_{latched}}$ with $\overline{\mathbf{P_{S=R}}}$. The reason for using an inverted $\mathbf{P_{S=R}}$, is because $\mathbf{P_{S=R}}$ is represented with inverted values. The output of the first NAND gate is therefore **0** only when **G** is **1** and $\mathbf{P_{S=R}}$ is carry-sensitive. The second NAND gate used, combines all three possible carry signals. The three signals are: a propagating carry-in, a generated carry from the first adder element or a generated carry from the second adder element. The output of this NAND gate is therefore the carry-out of the two first adder elements. The third NAND gate used (NAND 3), combines the **S** = **R** signal from adder elements three and four, with the inverted generated carry from adder element one. If **S** = **R** of adder element three and four is carry sensitive, this is represented with a logic **1**. Therefore, to obtain the desired functionality, the inverted generated carry from adder element one needs to be inverted once more. The output of the third NAND gate is **0** only when the first adder element generates a carry, and the second, third and fourth adder elements are carry-sensitive. The fourth NAND gate (NAND 4) combines the generated carry from adder element one, with the generated carry signals from adder elements two, three, four and an eventual propagating carry-in. Thus the output of the fourth NAND gate is the carry-out of the four first adder elements. The fifth NAND gate used (NAND 5) combines the generated carry from adder element one with the two remaining **S** = **R** signals, from adder elements five and six, and seven and eight respectively. The functionality is equal to NAND 3, therefore the

generated carry needs to be inverted once more. The output of fifth NAND gate is **0** only when the remaining **S** = **R** are carry sensitive, and a generated carry has passed the aforementioned gates. In the sixth NAND gate to be described, all generated carry signals are combined along with an eventual propagating carry-in. If any of the inputs of this NAND gate are **0**, a carry-out is generated. The mentioned example counts for the other generated carry signals as well.

It is worth noticing that the gate delay of the generated carry from adder element one, is $6\Delta_G$. Also, the worst gate-delay in the CLA Generator, the path of the generated carry from adder element one to the carry-in on the eight adder element, is $9\Delta_G$. Although a complicated structure, the gate-delay is still smaller than for the 16-bit MV full-adder. Time constraints has been mentioned as one of the reasons why the proposed design has not been implemented in either schematics or layout. Another reason is that due to the latching in the adder elements, gate-delay calculations on the generated carry signals was considered to be unimportant. Furthermore, the figure also illustrates that the proposed solution increases the number of gates needed dramatically. If this solution was to be implemented as a 32-bit adder, the complexity would be even greater. But since the CLA Generator makes use of binary gates, it will sustain high frequency. This will therefore not pose any delay for the adder elements used. A better solution concerning handling of the generated carry signal may still be desired. It is therefore important to point out that this is merely one proposed solution of the design.

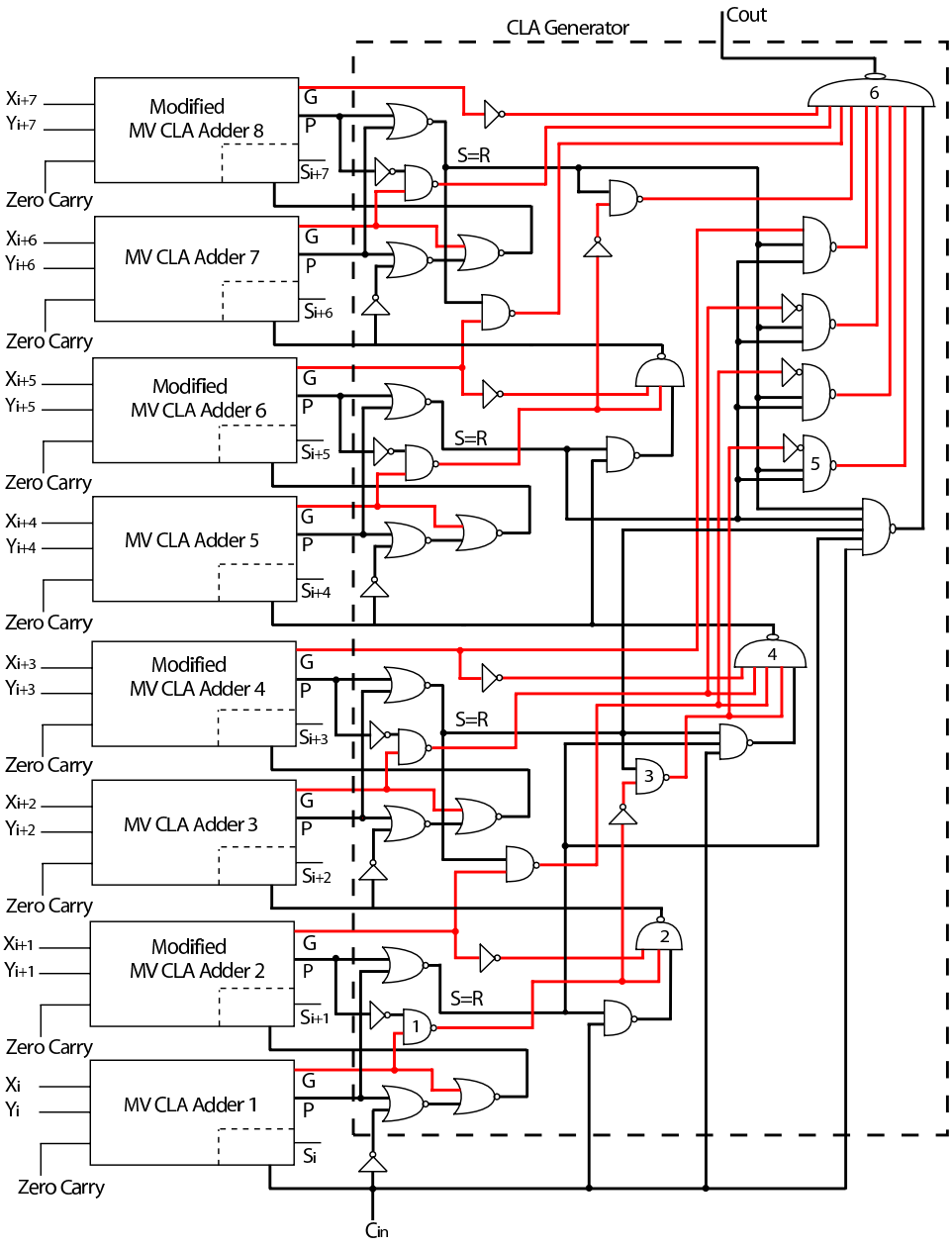

Figure B.1: *The figure illustrates a proposal of a complete 16-bit MV CLA full-adder.* **P** *and* **G** *represent* $\mathbf{P_{S=R}}$ *and* $\mathbf{G_{latched}}$.

Appendix C

Transistor tables

The first table represents the transistor sizes of the implemented MV CLA full-adder, while the second table represents the transistor sizes of the implemented MV full-adder.

SFG Inverter nr	W nMOS	W pMOS
1	0.6µm	3.05µm
2	1.8µm	10.0µm
3	0.6µm	3.05µm
4	0.6µm	3.05µm
5	0.6µm	3.05µm
6	0.6µm	3.05µm
7	0.6µm	3.05µm
8	0.6µm	3.05µm
9	1.8µm	10.0µm
10	0.6µm	3.05µm
11	0.6µm	3.05µm
NAND 1	10.0µm	10.0µm
NOR 1	2.0µm	6.1µm
NOR1	2.0µm	6.1µm
Bin Inv	0.6µm	3.05µm

Figure C.1: *Width of the transistors used for the implementation of the MV CLA full-adder. All transistors have a length of 0.35 µm. Furthermore, the recharge transistors have a W/L of 1.25/0.35 µm and 1.3/0.35 µm for the nMOS and pMOS transistors respectively.*

SFG Inverter nr	W nMOS	W pMOS
1	0.6μm	3.05μm
2	1.8μm	10.0μm
3	0.6μm	3.05μm

Figure C.2: *Width of the transistors used for the implementation of the MV full-adder. All transistors have a length of 0.35 μm. Furthermore, the recharge transistors have a W/L of 1.25/0.35 μm and 1.3/0.35 μm for the nMOS and pMOS transistors respectively.*

Appendix D

Instruments and Pin-out overview

D.1 Test Setup

To be able to measure the prototyp chip, several instruments were needed. The instruments used are found in Table D.1, with a description of their functionality. Furthermore, a schematic map of the test setup including the prototype chip, is presented in Figure D.1.

Instrument	Description
Hewlett Packard E3614A	DC power supply
Hewlett Packard E3610A	DC power supply
Hewlett Packard 54503	Oscilloscope
TTi TGA1244	Waveform generator

Table D.1: *Instruments used for measurements.*

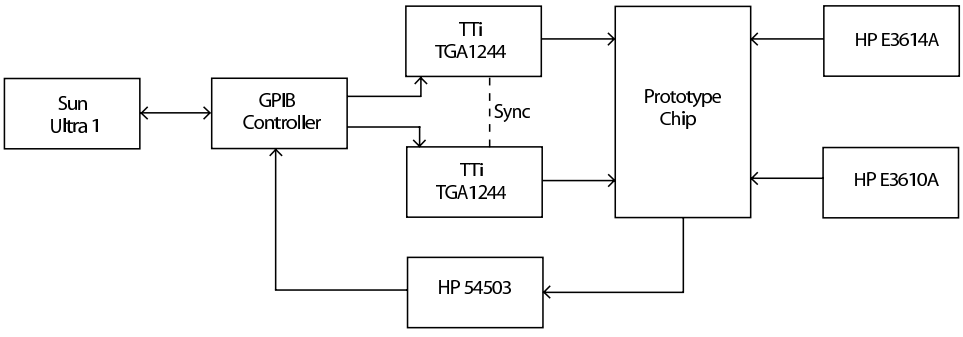

Figure D.1: *Map of the test network used for verification of the prototype chip.*

D.2 Pin-out overview

To provide easy access for measurements on the prototype chip, pin-out maps are presented. The pin-out maps provide the input/output name and the geographic position of these on the finished chip. Pin-out maps to both prototype circuits are provided.

Test Adder Description	I/O	Pin Out
PAD Vss		M5
PAD Vdd		N5
Vss		A2
Vdd		B3
CLK'		A1
CLK		B2
Zero-Carry	INClk	B1
Carry-In	Cin	C2
Bit1	In1	C1
Bit2	In2	D2
Bit3	In3	D1
Bit4	In4	E2
Output of AutoZero of Bit3	Out	E1
Output of BMVC using Bit3 & Bit4 as inputs	Out	F3
Output of BMVC using Bit1 & Bit2 as inputs	Out	F2
Node Z	Out	F1
Output of G	Out	G2
Output of P	Out	G3
Output of $\overline{G_{latched}}$	Out	G1
$G_{latched}$	Out	H1
$P_{S=R}$	Out	H2
Output of Carry-In inverted	Out	H3
C_{prop}	Out	J1
$G_{latched}$	Out	J2
$\overline{C_{Out}}$	Out	K1
Output of Bit1	Out	K2
Output of Bit0	Out	L1
$Sum_{latched}$	Out	L2
Output of R4 inverter 2	Sum	M1
G (adder element)	Out	N1
Q	Out	M2
Output of Carry-In after AutoZero	Out	N2

Table D.2: *Pin List for single adder, test element.*

4 Comined Adders: Description	I/O	Pin Out
PAD Vss		M5
PAD Vdd		N5
Vss		A2
Vdd		B3
CLK'		B4
CLK		A3
Zero-Carry	INClk	A41
Carry-In	Cin	B11
Bit1	In1	A11
Bit2	In2	B10
Bit3	In3	A10
Bit4	In4	B9
Bit5	In5	A9
Bit6	In6	C8
Bit7	In7	B8
Bit8	In8	A8
Bit9	In9	B7
Bit10	In10	C7
Bit11	In11	A7
Bit12	In12	A6
Bit13	In13	B6
Bit14	In14	C6
Bit15	In15	A5
Bit16	In16	B5
Output Bits	Bit0	A12
	Bit1	B12
	Bit2	C12
	Bit3	B13
	Bit4	D12
	Bit5	D13
	Bit6	E13
	Bit7	F11
Carry-Out of each element	Cout1	A13
	Cout2	C13
	Cout3	E12
	Cout4	F12

Table D.3: *Pin List for cascaded adders.*

Appendix E

Additional Figures

.tex In this section, additional layout figures are presented. Since most circuits used, resemble the two-input inverter presented in chapter four, only completely different designs are presented here. Furthermore, a plot of the prototype chip with pads is provided.

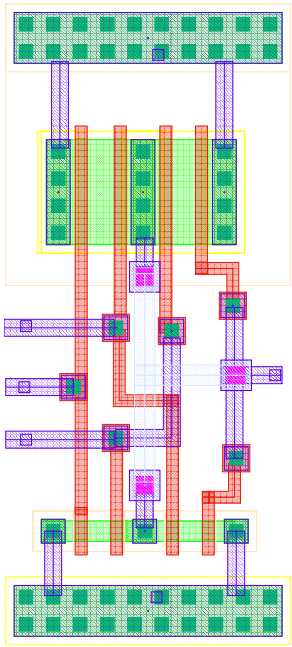

Figure E.1: *The layout design of the AutoZero circuits used.*

.tex

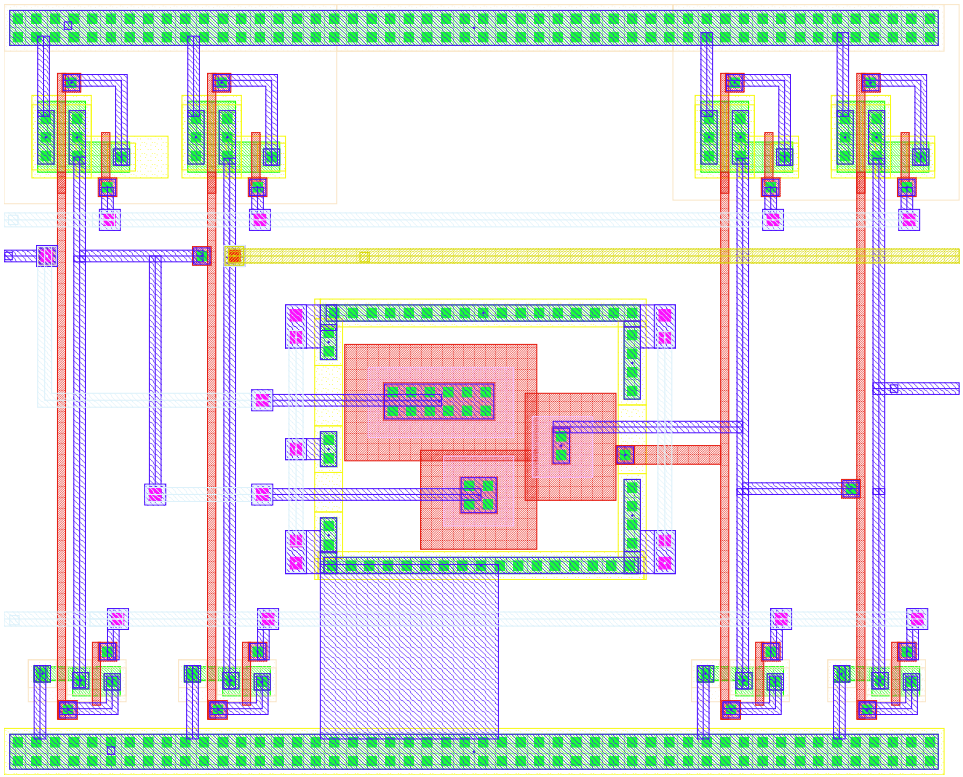

Figure E.2: *The layout design of the Binary-to-Multiple-Valued Converter used for the prototype chip.*

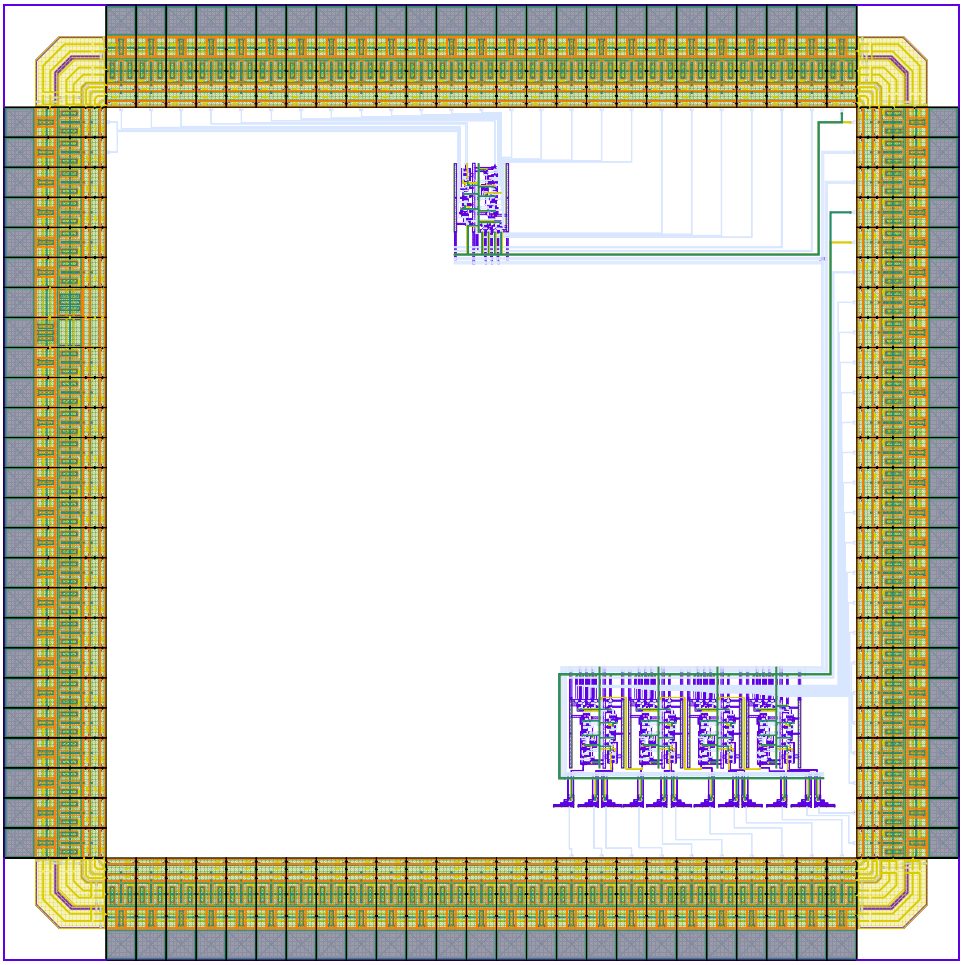

Figure E.3: *The layout design of the prototype chip. Notice that the clock signals are routed on-top of each other. The empty pads are used by another circuit, not present on this illustration.*

Appendix F

Matlab scripts

F.1 Scripts

In the following the two sentral matlab scripts that were used are included.

F.1.1 Generating inputs and clock signals with the TTi instruments

```
%% Reset instruments
TTi1244A = TTi1244_DefaultName;
GPIB_Write('*CLS;*RST;',TTi1244A)
TTi1244B = '/dev/TTi1244B';
GPIB_Write('*CLS;*RST;',TTi1244B)

HP54503=HP54503_Defaultname;


bunn = [-2048];
topp = [2047];
antpkt = 64*input('Antall 64x punkter\n');

% Frequency is 1/8 of f:
f=4e3;

CLKinv = [];
klokke = [];
for j=1:antpkt/4,
  for i=1:antpkt/32,
    klokke =[klokke bunn];
    CLKinv = [CLKinv topp];
  end
  for i=1:antpkt/32,
```

```
    klokke = [klokke topp];
    CLKinv = [CLKinv bunn];
  end
end


Inn1 = [];
for j=1:antpkt/8,
  for i=1:antpkt/16,
    Inn1 =[Inn1 bunn];
  end
  for i=1:antpkt/16,
    Inn1 = [Inn1 topp];
  end
end


Inn2 = [];
for j=1:antpkt/16,
  for i=1:antpkt/8,
    Inn2 =[Inn2 bunn];
  end
    for i=1:antpkt/8,
    Inn2 = [Inn2 topp];
  end
end


Inn3 = [];
for j=1:antpkt/32,
  for i=1:antpkt/4,
    Inn3 =[Inn3 bunn];
  end
  for i=1:antpkt/4,
    Inn3 = [Inn3 topp];
  end
end


Inn4=[];
for j=1:antpkt/64,
  for i=1:antpkt/2,
    Inn4 =[Inn4 bunn];
  end
  for i=1:antpkt/2,
    Inn4 = [Inn4 topp];
  end
end
```

```
% setup
GPIB_Write('EOI ON', HP54503);
GPIB_Write('DISPLAY:CONNECT ON', HP54503);
GPIB_Write('BNC PROBE', HP54503);

GPIB_Write('SETUPCH 1',TTi1244A);
GPIB_Write('SYNCOUT ON', TTi1244A);



TTi1244_LockStatus('OFF');

TTi1244_SetChannel(1);
TTi1244_ArbitraryChannelDelete('vin4');
TTi1244_ArbitraryBackDelete('vin4');
TTi1244_ArbitraryDef('vin4', length(Inn4), Inn4);
TTi1244_ArbitrarySetOutput('vin4');
TTi1244_SetTerm('OPEN');
TTi1244_LockMode('MASTER');
TTi1244_ArbitraryFrequency(f);
TTi1244_SetAmplitude(2);
TTi1244_DCoffset(1.0281);
TTi1244_ChannelEnable('ON', 1);

TTi1244_SetChannel(2);
TTi1244_ArbitraryChannelDelete('vin3');
TTi1244_ArbitraryBackDelete('vin3');
TTi1244_ArbitraryDef('vin3', length(Inn3), Inn3);
TTi1244_ArbitrarySetOutput('vin3');
TTi1244_SetTerm('OPEN');
TTi1244_LockMode('SLAVE');
TTi1244_ArbitraryFrequency(f);
TTi1244_SetAmplitude(2);
TTi1244_DCoffset(1.0281);
TTi1244_ChannelEnable('ON', 2);


TTi1244_LockStatus('ON');

TTi1244_LockStatus('OFF',TTi1244B);

TTi1244_SetChannel(1, TTi1244B);
TTi1244_ArbitraryChannelDelete('vin2', TTi1244B);
TTi1244_ArbitraryBackDelete('vin2', TTi1244B);
TTi1244_ArbitraryDef('vin2', length(Inn2), Inn2, TTi1244B);
TTi1244_ArbitrarySetOutput('vin2', TTi1244B);
TTi1244_SetTerm('OPEN', TTi1244B);
TTi1244_LockMode('MASTER', TTi1244B);
TTi1244_ArbitraryFrequency(f, TTi1244B);
```

```
TTi1244_SetAmplitude(2, TTi1244B);
TTi1244_DCoffset(1.0281, TTi1244B);
TTi1244_ChannelEnable('ON', 1, TTi1244B);


TTi1244_SetChannel(2, TTi1244B);
TTi1244_ArbitraryChannelDelete('vin1', TTi1244B);
TTi1244_ArbitraryBackDelete('vin1', TTi1244B);
TTi1244_ArbitraryDef('vin1', length(Inn1), Inn1, TTi1244B);
TTi1244_ArbitrarySetOutput('vin1', TTi1244B);
TTi1244_SetTerm('OPEN', TTi1244B);
TTi1244_LockMode('SLAVE', TTi1244B);
TTi1244_ArbitraryFrequency(f, TTi1244B);
TTi1244_SetAmplitude(2, TTi1244B);
TTi1244_DCoffset(1.0281, TTi1244B);
TTi1244_ChannelEnable('ON', 2, TTi1244B);

TTi1244_SetChannel(3, TTi1244B);
TTi1244_ArbitraryChannelDelete('klokke', TTi1244B);
TTi1244_ArbitraryBackDelete('klokke', TTi1244B);
TTi1244_ArbitraryDef('klokke', length(klokke), klokke, TTi1244B);
TTi1244_ArbitrarySetOutput('klokke', TTi1244B);
TTi1244_SetTerm('OPEN', TTi1244B);
TTi1244_LockMode('SLAVE', TTi1244B);
TTi1244_ArbitraryFrequency(f, TTi1244B);
TTi1244_SetAmplitude(2, TTi1244B);
TTi1244_DCoffset(1.0281, TTi1244B);
TTi1244_ChannelEnable('ON', 3, TTi1244B);


TTi1244_SetChannel(4, TTi1244B);
TTi1244_ArbitraryChannelDelete('CLKinv', TTi1244B);
TTi1244_ArbitraryBackDelete('CLKinv', TTi1244B);
TTi1244_ArbitraryDef('CLKinv', length(CLKinv), CLKinv, TTi1244B);
TTi1244_ArbitrarySetOutput('CLKinv', TTi1244B);
TTi1244_SetTerm('OPEN', TTi1244B);
TTi1244_LockMode('SLAVE', TTi1244B);
TTi1244_ArbitraryFrequency(f, TTi1244B);
TTi1244_SetAmplitude(2, TTi1244B);
TTi1244_DCoffset(1.0281, TTi1244B);
TTi1244_ChannelEnable('ON', 4, TTi1244B);


TTi1244_LockStatus('OFF',TTi1244B);



GPIB_Write('REFCLK MASTER',TTi1244A);
GPIB_Write('REFCLK SLAVE',TTi1244B);
```

```
TTi1244_LockStatus('ON',TTi1244B);

input('Prepare scope for readout');
DumpScope;

figure(2);
hold off;
plot(channelX(1,:), channelY(1,:)); grid on;
```

F.1.2 Screen Dump from the HP54503A

```
GPIB_Write('EOI ON', HP54503);
GPIB_Write('DISPLAY:CONNECT ON', HP54503);
GPIB_Write('BNC PROBE', HP54503);

auto = 0;
% auto = input('Autoscale and reset the scope? (1/0)\n');
if auto == 1,
  GPIB_Write('*RST', HP54503);
  GPIB_Write('AUTOSCALE', HP54503);
  pause(10);
end

%GPIB_Write('DISPLAY:CONNECT ON', HP54503);
%GPIB_Write('BNC PROBE', HP54503);

format = [];
type = [];
nopoints = [];
count = [];
xinc = [];
xorg = [];
xref = [];
yinc = [];
yorg = [];
yref = [];

channelY = [];
channelX = [];

for channelno = 1 : 4
  cmdstr = sprintf('WAVEFORM:SOURCE CHANNEL%d', channelno);
  GPIB_Write(cmdstr, HP54503);
  GPIB_Write('ACQUIRE:COMPLETE 30', HP54503);
%  cmdstr = sprintf('DIGITIZE CHANNEL%d', channelno);
%  GPIB_Write(cmdstr, HP54503);
```

```
  GPIB_Write('SYSTEM:HEADER OFF', HP54503);
  GPIB_Write('WAVEFORM:PREAMBLE?', HP54503);
  data = GPIB_Read(HP54503);

% vectorize comma-separated data in string.
% data - input string
% tmp - vectorized data

  tmp = [0:9];
  k = 1;
  j = [];
  for i = 1 : length(data)
    if data(i) ~= ',',
      j = [j data(i)];
    end
    if data(i) == ',',
      tmp(k) = str2num(j);
      k = k + 1;
      j = [];
    end
  end
  %tmp(k) = str2num(j);


  format(channelno) = tmp(1);
  type(channelno) = tmp(2);
  nopoints(channelno) = tmp(3);
  count(channelno) = tmp(4);
  xinc(channelno) = tmp(5);
  xorg(channelno) = tmp(6);
  xref(channelno) = tmp(7);
  yinc(channelno) = tmp(8);
  yorg(channelno) = tmp(9);
  yref(channelno) = tmp(10);


  GPIB_Write('WAVEFORM:FORMAT ASCII', HP54503);
  GPIB_Write('WAVEFORM:DATA?', HP54503);
  b = GPIB_Read(HP54503);

  array = [1:500];
  k = 1;
  j = [];
  for i = 1 : length(b)
    %b(i)
    if b(i) ~= ','
      j = [j b(i)];
    end
    if b(i) == ','
```

```
      array(k) = str2num(j);
      k = k + 1;
      j = [];
    end
  end

  %plot(array)

  y = array;

  for i = 1 : length(y)
    y(i) = (y(i) - yref(channelno))*yinc(channelno) + yorg(channelno);
  end

  channelY(channelno, :) = y;


  x = [0:499];
  for i = 1 : length(y)
    x(i) = (x(i) - xref(channelno))*xinc(channelno) + xorg(channelno);
  end

    channelX(channelno, :) = x;


%  figure(channelno*10000);
%  plot(x, y); grid on;

end

figure(100);
subplot(4,1,1), plot(channelX(1,:), channelY(1,:)); grid on;
subplot(4,1,2), plot(channelX(2,:), channelY(2,:)); grid on;
subplot(4,1,3), plot(channelX(3,:), channelY(3,:)); grid on;
subplot(4,1,4), plot(channelX(4,:), channelY(4,:)); grid on;

%axis([-0.025 0.025 -0.1 2.1]);
date = clock;
date2 = sprintf('%d.%d.%d', date(3), date(2), date(1));
ylabel('Out (V)');
```

Appendix G

Additional Simulations

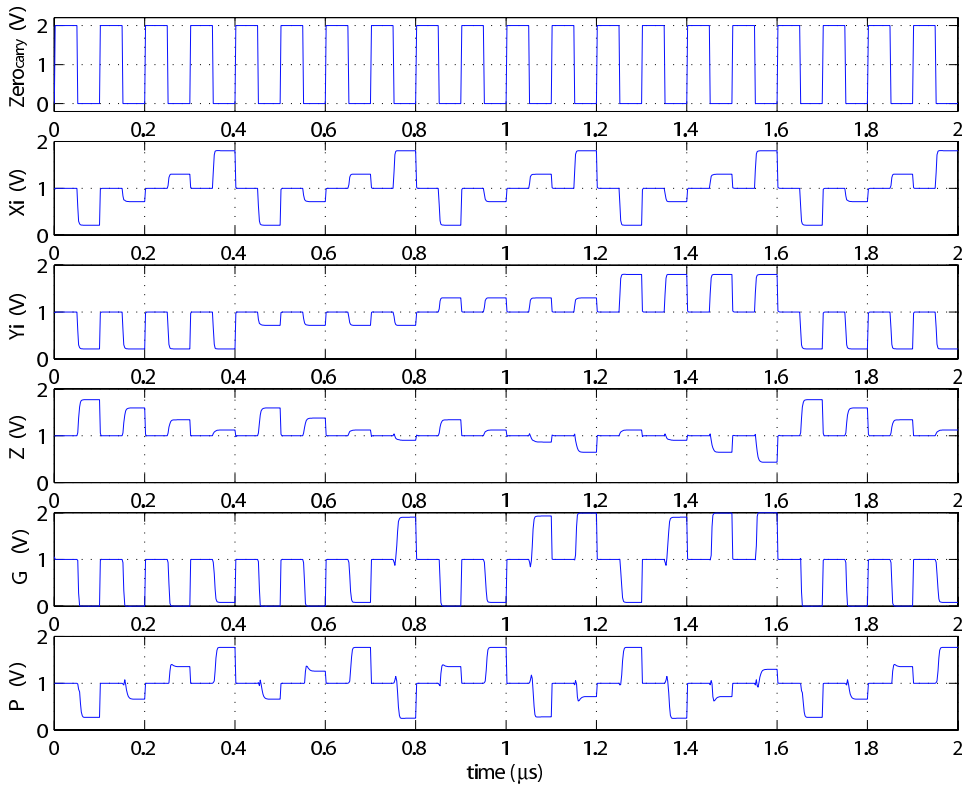

Figure G.1: *Layout simulation of the the carry element of the adder. The* $\mathbf{Zero_{Carry}}$ *is equal to the clock signal used. The* $\mathbf{Zero_{Carry}}$, $\mathbf{X_i}$ *and* $\mathbf{Y_i}$ *are summed, resulting in the R8 signal (***Z***). The R8 signal is used to determine the internal carry (***G***), which in turn is summed with the latter to give us the R4 signal (***P***). The frequency is 10MHz.*

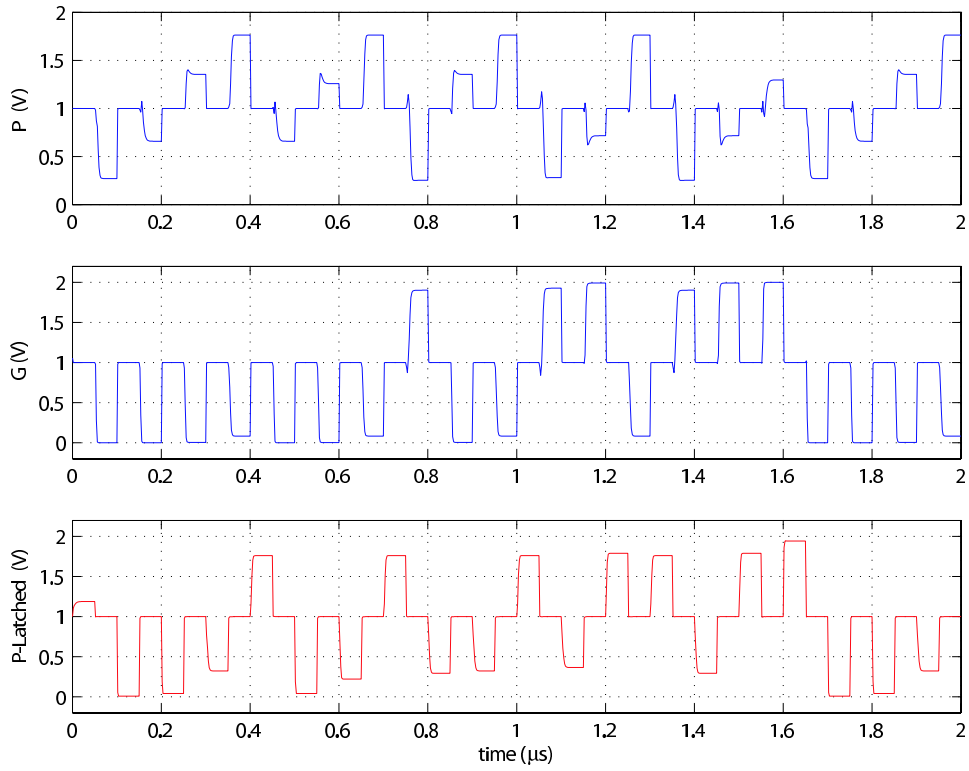

Figure G.2: *Layout simulation of the carry element of the adder. The internal carry (***G***) is latched with the R4 signal (***P***), resulting in the latched representation of (***P***), namely (*$\mathbf{P_{latched}}$*). The frequency is 10MHz.*

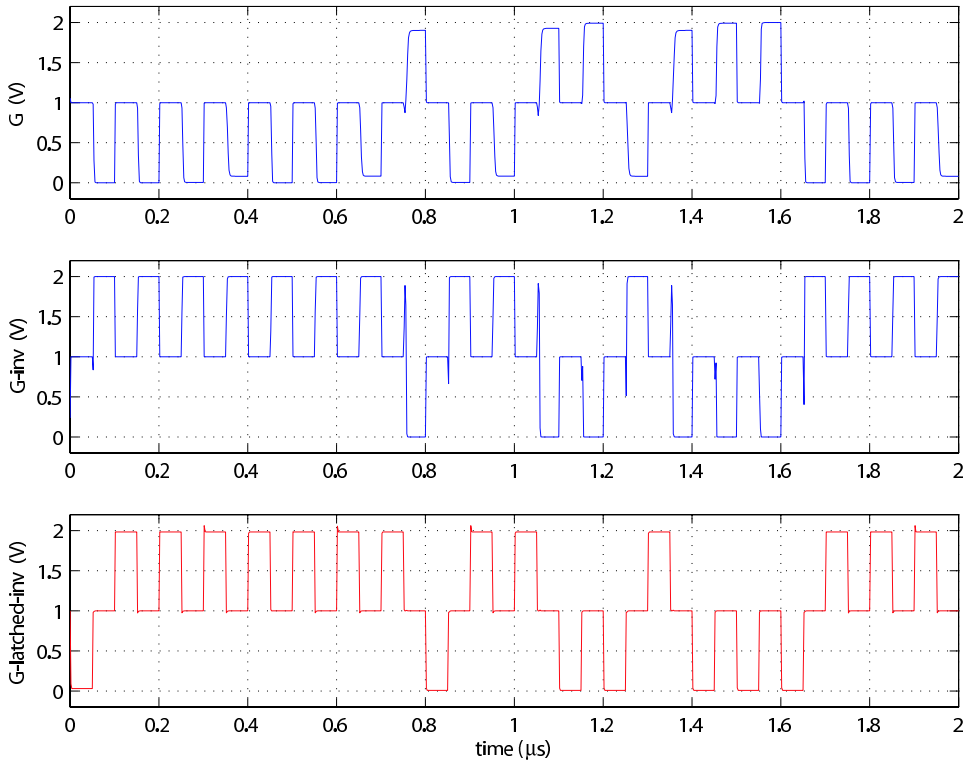

Figure G.3: *Layout simulation of the carry element of the adder. The internal carry (*$\mathbf{G}$*) is inverted (*$\overline{\mathbf{G}}$*), and then latched. The frequency is 10MHz.*

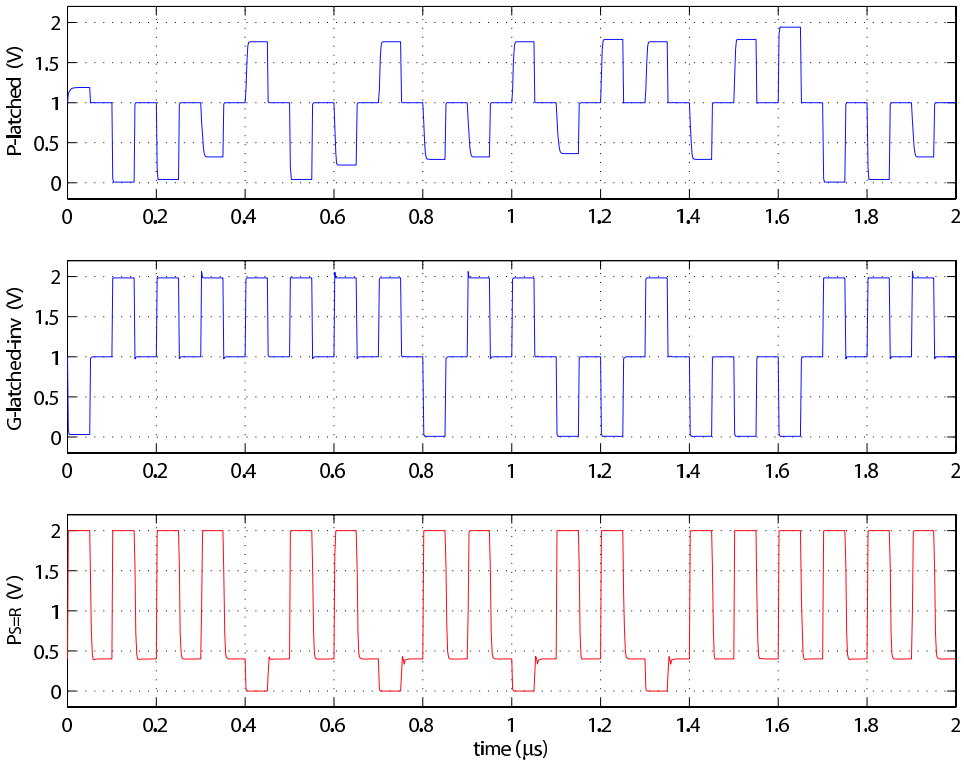

Figure G.4: *Layout simulation of the carry element of the adder.* $\mathbf{P_{latched}}$ *and the latched* $\overline{\mathbf{G}}$ *are used as inputs on the NAND gate, resulting in the carry-in sensitive sum* $\mathbf{P_{S=R}}$*. The frequency is 10MHz.*

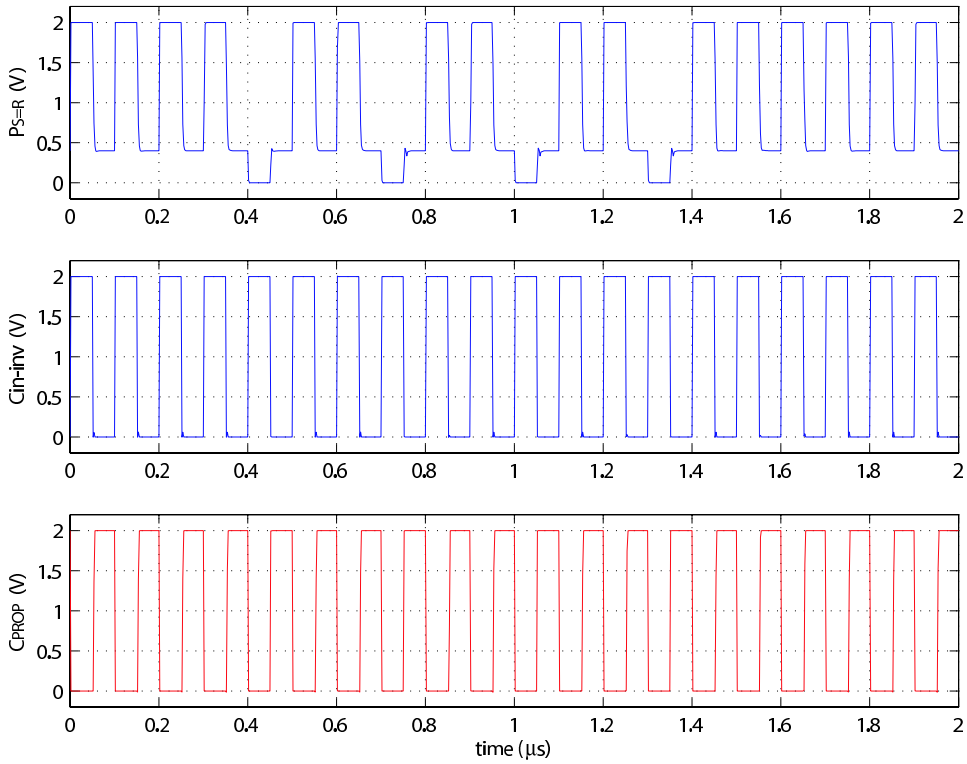

Figure G.5: *Layout simulation of the carry element of the adder. The output of the NAND gate (*$\mathbf{P_{S=R}}$*) and the inverted* $\mathbf{Carry_{in}}$ *signals are used as inputs on the NOR gate, resulting in the propagating carry (*$\mathbf{C_{PROP}}$*). In this simulation the* $\mathbf{Carry_{in}}$ *is zero. The frequency is 10MHz.*

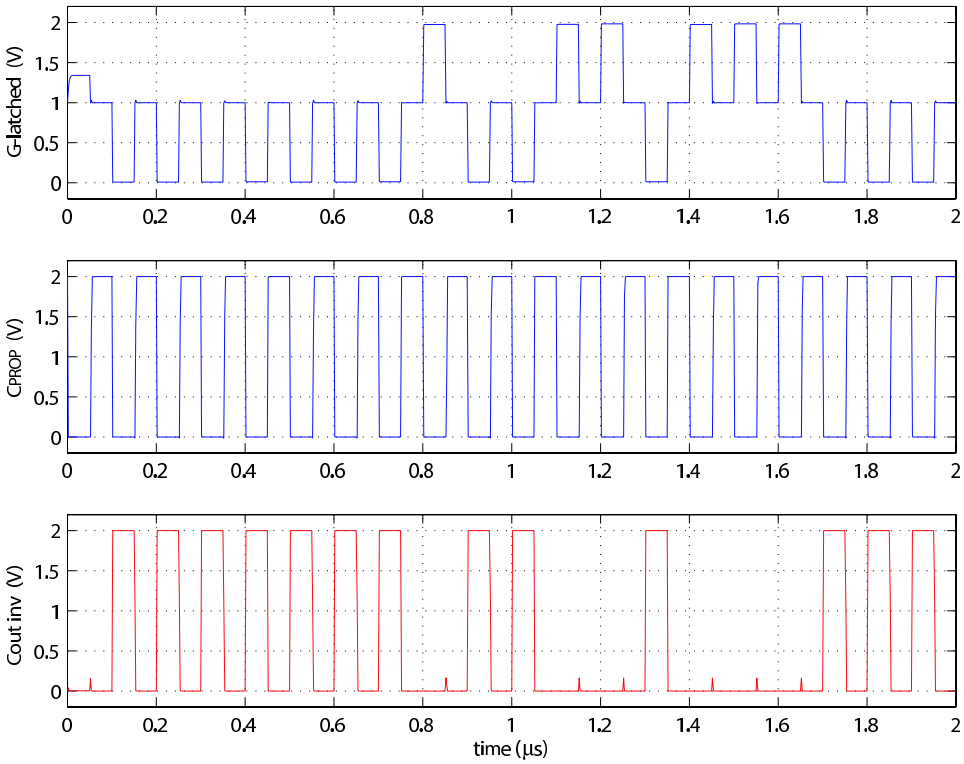

Figure G.6: *Layout simulation of the carry element of the adder. The output of the NOR gate (*$\mathbf{P_{S=R}}$*) and* $\overline{\mathbf{G}}$ *are used as inputs on a second NOR gate. The output of this NOR gate is an inverted Carry out signal (*$\overline{\mathbf{C}}_{\mathbf{out}}$*). This simulation shows the carry out when the carry in is logic* **0**. *The frequency is 10MHz.*

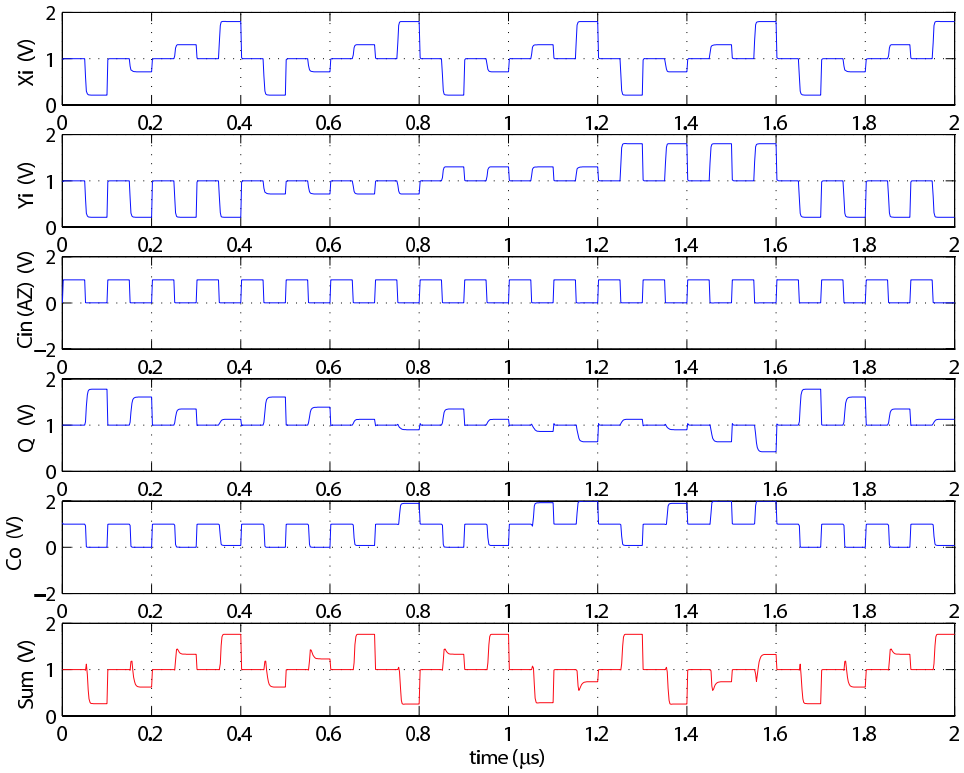

Figure G.7: *Simulation of layout, showing how the adder element operates. The* $\mathbf{Carry_{in}}$ *is logic* **0** *in this simulation. The two R4 signals are generated by two BMVC's. Node* **Q** *represents the the radix-8 sum of the inputs, while* **G** *represents the* **Carry** *signal, though only used to calculate the output,* **Sum**. *The frequency is 10MHz.*

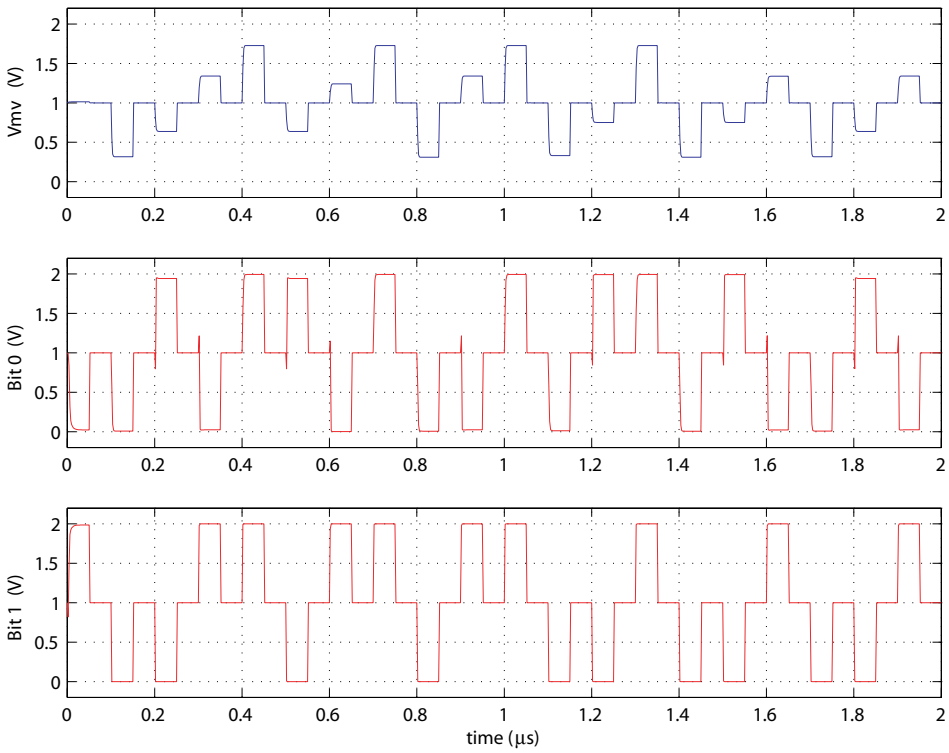

Figure G.8: *Simulation of layout, showing how the MVBC operates. The R4 input results in two a two-bit output, the output bits are represented with separate wires. In this simulation the carry in is logic* **0**. *The frequency is 10MHz.*

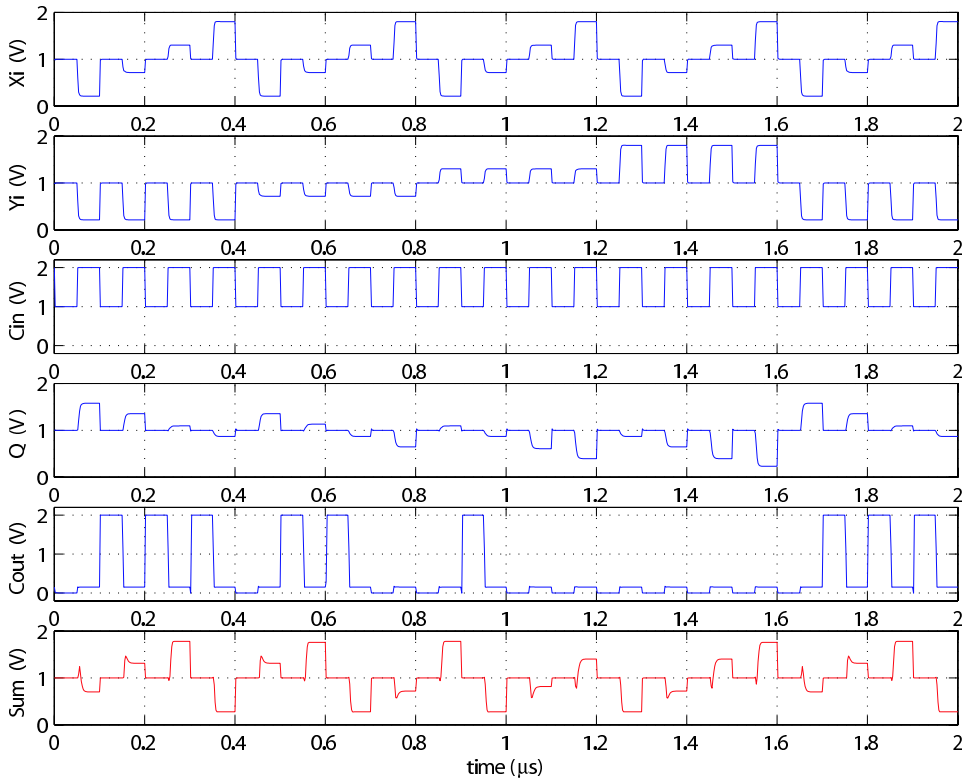

Figure G.9: *Simulation of layout, showing how the adder element operates. The* $\mathbf{Carry_{in}}$ *is logic* **1** *in this simulation. The two R4 signals are generated by two MVBC's. Node* **Q** *represents the the radix-8 sum of the inputs, while* **G** *represents the* **Carry** *signal, though only used to calculate the output,* **Sum**. *The frequency is 10MHz.*

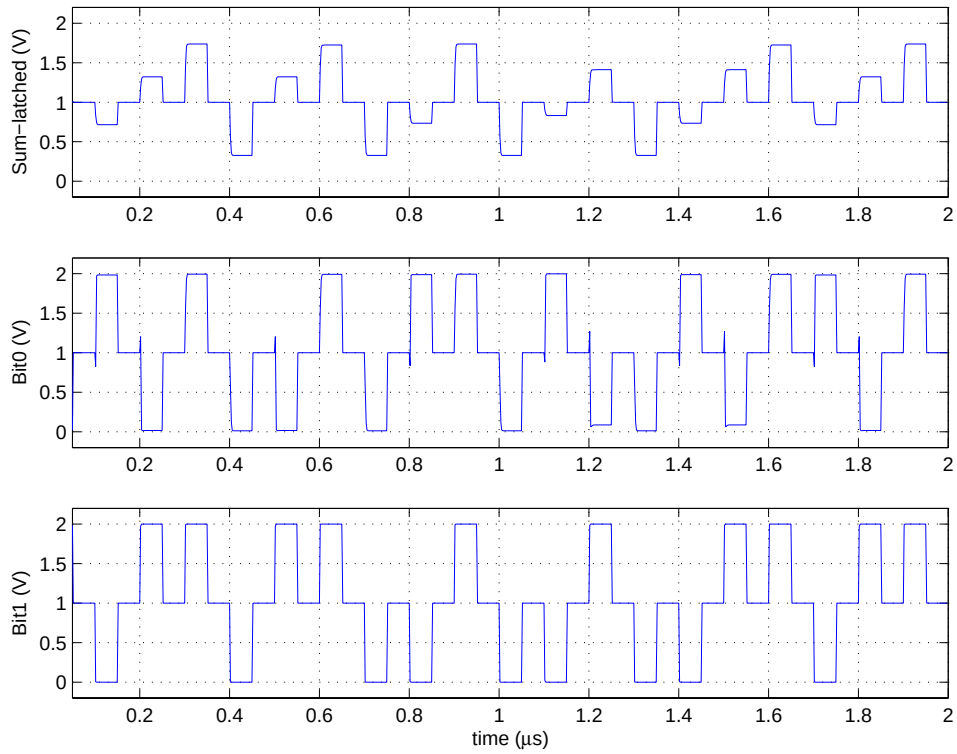

Figure G.10: *Simulation of layout, showing how the MVBC operates. The R4 input results in two bit output, the output bits are represented with separate wires. In this simulation the carry in is logic* **1**. *The frequency is 10MHz.*

Appendix H

Glossary

AMS	Austria Mikrosysteme
AZ	Auto-Zero
BMVC	Binary-to-Multiple-Valued Converter
CLA	Carry-Look-Ahead
CMOS	Complementary Metal-Oxide-Silicon
EDP	Energy-Delay-Product
FG	Floating-Gate
LSB	Least Significant Bit
MOS	Metal-Oxide-Silicon
MOSFET	Metal-Oxide-Silicon Field Effect Transistor
MSB	Most Significant Bit
MV	Multiple-Valued
MVBC	Multiple-Valued-to-Binary Converter
MVL	Multiple-Valued Logic
PDP	Power-Delay-Product
SFG	Semi-Floating-Gate
Tm	Typical-mean
UVFG	Ultra-Violet Floating-Gate
Wo	Worst one
Wz	Worst zero

Bibliography

[1] I. Koren **"Computer Arithmetic Algorithms"** , A. K. Peters Ltd, Second edition, 2002, pp. 95-99 & pp. 106-109.

[2] K. Raahemifar, and M. Ahmadi, **"Fast Carry-Look-Ahead Adder"**, Proceedings of the 1999 IEEE Canadian Conference on Electrical and Computer Engineering, pp. 529-532.

[3] G. A. Ruiz, and M. A. Manzano, **"Compact 32-bit CMOS Adder in Multiple-Output DCVS Logic for Self-Timed Circuits"**, IEE Proc.-Circuits Devices Syst., vol. 147, no. 3, June 2000.

[4] Y. T. Lee, I. C. Park and C. M. Kyung, **"Design of Compact Static CMOS Carry Look-Ahead Adder using Recursicve Output Property"**, Electronics Letters 29th April 1993, vol. 29, no9 pp. 794-796.

[5] K. Ueda, H. Suzuki, K. Suda, H. Shinohara and K. Mashiko, **"A 64-bit Carry Look-Ahead Adder using Pass Transistor BiCMOS Gates"**, IEEE Journal of Solid-State Circuits, vol. 31, no. 6, pp. 810-818, June 1996.

[6] K. Ueda, H. Suzuki, K. Suda, H. Shinohara and K. Mashiko, **"A 64-bit Carry Look-Ahead Adder using Pass Transistor BiCMOS Gates"**, IEEE Journal of Solid-State Circuits, vol. 31, no. 6, June 1996.

[7] H. Y. Huang, and T. N. Wang, **"High-Speed CMOS Logic Circuits in Capacitor Coupling Technique"**, The 2001 IEEE International Symposium on Circuits and Systems, vol. 4, pp. 634-637, 6-9 May 2001, ISCAS 2001.

[8] Tadashi Shibata and Tadahiro. Ohmi, **"A Functional MOS Transistor Featuring Gate-Level Weighed Sum and Threshold Operations"**, IEEE Transactions on Electron Devices, vol. 39, no. 6. pp 1444-1455, June 1992.

[9] Tadashi Shibata and Tadahiro. Ohmi, **"Neuron MOS Binary-Logic Integrated Circuits-Part I: Design Fundamentals and Soft-Hardware-Logic Circuit Implementation"**, IEEE Transactions on Electron Devices, vol. 40, no. 3. pp 570-576, March 1993.

[10] Tadashi Shibata and Tadahiro. Ohmi, **"Neuron MOS Binary-Logic Integrated Circuits-Part II: Simplifying Tecniques of Circuit Configuration and their Practical Applications"**, IEEE Transactions on Electron Devices, vol. 40, no. 5. pp 974-979, May 1993.

[11] J. Shen, K. Tanno, O. Ishizuka and Z. Tang, **"Neuron-MOS current mirror circuit and its application to multi-valued logic"** , IEICE, Vol. E82-D, No. 5, pp. 940-948, May 1999.

[12] Y. Berg, T.S. Lande, Ø. Næss, and H. Gundersen, **"Ultra-low-Voltage Floating-Gate Transconductance Amplifiers"**, IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing, Vol. 48, No. 1, pp. 37-44, January 2001.

[13] Y. Berg, O. Næss, and M. Høvin, **"Ultra lowvoltage floating-gate transconductance amplifier"**, ISCAS 2000, IEEE International Symposium in Circuits and Systems, May 28-31, 2000.

[14] Y. Berg, D. T. Wisland, and T. S. Lande, **"Ultra Low-Voltage/Low-Power Digital Floating-Gate Circuits"**, IEEE International Symposium in Circuits and Systems-II: Analog and Digital Signal Processing, vol. 46, no.7, July 1999.

[15] Y. Berg, T. S. Lande, O. Næss and H. Gundersen, **"Ultra-Low-Voltage Floating-Gate Transconductance Amplifiers"**, IEEE International Symposium in Circuits and Systems-II: Analog and Digital Signal Processing, vol. 48, no.1, January 2001.

[16] S. Aunet, Y. Berg and T. Sæther, **"Real-Time Reconfigurable Linear Threshold Elements Implemented in Floating-Gate CMOS"**, IEEE Transactions on Neural Networks , vol. 14, no.5, September 2003.

[17] Y. Berg, T. S. Lande and O. Næss, **"Programming Floating-Gate Circuits with UV-Activated Conductances"**, IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing, vol. 48, no. 1, January 2001.

[18] Koji Kotani, Tadashi Shibata, Makoto Imai and Tadahiro. Ohmi, **"Clock-Controlled Neuron-MOS Logic Gates"**, IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, vol. 45, no. 4. pp 518-522, April 1998.

[19] O. Mirmotahari and Y. Berg, **"A novel multiple-input multiple-valued semi-floating-gate latch"**, Proc. IEEE International Symposium on Circuits and Systems, vol. 2, pp. II-592–II-595, May, 2002.

[20] O. Mirmotahari, **"Novel Latching Scheme in Multiple-Valued Recharge Logic"**, Cand. Scient. Thesis, May, 2003.

[21] P. Hafliger and H.K. Riis, **"A Multi-Level Static Memory Cell"** , IEEE, International Symposium Circuits and Systems, 2004. ISCAS 2004, Vol. 5, pp 393-396, 23-26 May 2004.

[22] H.K. Riis, and P. Hafliger, **"Spike based learning with weak multi-level static memory"** , IEEE, Proceedings of the 2003 International Symposium Circuits and Systems. ISCAS 2003, Vol. 1, pp 25-28, May 2003.

[23] Paul R. Gray, and Robert G. Meyer, **"Analysis an design of Analog Integrated Circuits"** , John Wiley & Sons, Second edition, 1984.

[24] Y. Berg, S. Aunet, Ø. Næss, O. Mirmotahari and M. Høvin, **"Binary to Multiple-Valued Recharge Converter for Multiple-Valued CMOS Logic"**, The 16th European Conference on Circuits Theory and Design, ECCTD'03, Krakow, Poland.

[25] Y. Berg, S. Aunet, Ø. Næss, and O. Mirmotahari, **"Basic Multiple-Valued Functions using Recharge Logic"**, Proceedings of the 34th International Symposium on Multiple-Valued Logic (ISMVL'04), pp. 346-351, May 2004.

[26] N.H.E. Weste and K. Eshragian, **"Principles of CMOS Design - A System Perspective"**, Addison-Wesley Publishing Company, Second edition, 1994.

[27] B. A. Minch, **"Floating-Gate Techniques for Assessing Mismatch"**, ISCAS 2000 - IEEE International Symposium on Circuits and Systems, May 28-31 2000, vol. 4, pp 385-388.

[28] K. R. Lakshmikumar, R. A. Hadaway and M. A. Copeland, **"Characterization and Modeling of Mismatch in CMOS Transistors for Precicion Analog Design"**, IEEE Journal of Solid-State Circuits, vol. sc-21, no. 6, pp. 1057-1066, December 1986.

[29] J. Bastos, M. Steyaert, B. Graindourze and W. Sansen, **"Matching of MOS Transistors with Different Layout Styles"**, Proceedings of the 1996 IEEE International Conference on Microelectronic Test Structures, vol. 9, pp. 17-18, March 1996.

[30] J. Pangjun, and S. S. Sapatnekar, **"Low-Power Clock Distribution Using Multiple Voltages and Reducing Swings"**, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 10, no. 3, pp 309-318, June 2002.

[31] A. Hastings **"The Art of Analog Layout"** , Prentice Hall Inc, 2001, Chapter 6.

List of Figures

List of Tables